Beyond CS106A

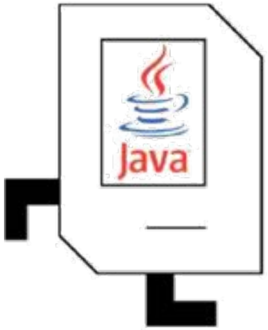
Chris Piech

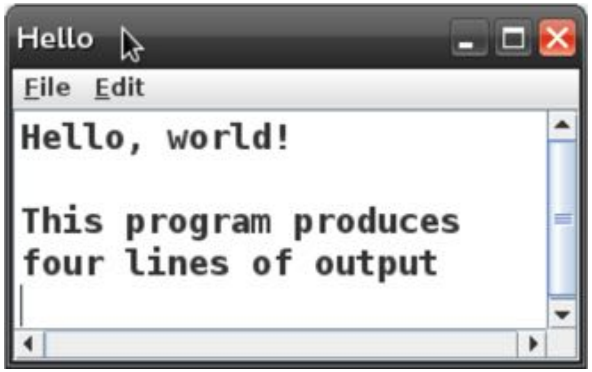
CS106A, Stanford University

Life after ACM

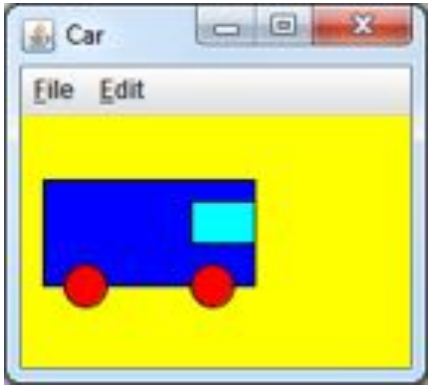
Life After The ACM Libraries

- All quarter we have relied on the **ACM Java libraries**.
  - Karel, ConsoleProgram, RandomGenerator
  - GraphicsProgram, GOval, GRect, GOval, GLine, GImage, ...





```
Hello, world!

This program produces
four lines of output
```



Click to start new game

Lives: 3 Score: 0

- Today we will see how **standard Java** programs are made.

Using the ACM Libraries

```
import acm.program.*;

public class MyProgram extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}
```

- This is a console program written using the ACM libraries.
  - It uses the **ConsoleProgram** class to represent a console.
  - The **run** method contains the program code.
  - The **println** method prints output to the graphical console.

A Barebones Java Program

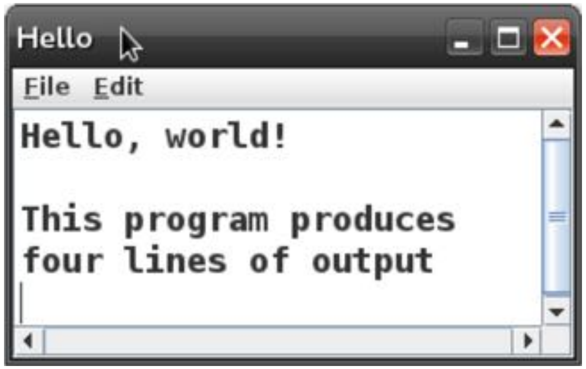
```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- The method **`main`** is the true entry point for a Java program.
  - It must have the exact heading shown above.
  - The `String[]` **`args`** are "command line arguments" (ignored).
  - The **`println`** command's true name is `System.out.println`.
  - Standard Java methods are **`static`** unless part of a class of objects.

Lets make Hello World!

Console Programs

- What does the **`ConsoleProgram`** library class do?
  - Creates a new graphical **window**
  - Puts a scrollable **text area** into it
  - Provides `print` and `println` commands to send text **output** to that window
  - contains a **`main` method** that calls your program class's `run` method
    - `ConsoleProgram`'s `run` is empty, but you extend and override it



```
Hello, world!

This program produces
four lines of output
```

```
public class Hello extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}
```

ACM console input

```
public class Age extends ConsoleProgram {
    public void run() {
        String name = readLine("What's your name? ");
        int age = readInt("How old are you? ");
        int years = 65 - age;
        println(name + " has " + years
                + " years until retirement!");
    }
}
```

- The ACM library has simple console input commands like `readLine`, `readInt`, `readDouble`, and so on.
- These methods display a 'prompt' message, wait for input, re-prompt if the user types a bad value, and return the input.

Java console input

```
public class Age {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("What's your name? ");
        String name = console.nextLine();
        System.out.print("How old are you? ");
        int age = console.nextInt();
        int years = 65 - age;
        System.out.println(name + " has " + years
                + " years until retirement!");
    }
}
```

- In standard Java, you must create a `Scanner` or similar object to read input from the console, which is also called `System.in`.
  - It does not automatically re-prompt and can crash on bad input.

Graphics Programs

The ACM library does several things to make graphics easier:

- Automatically creates and displays a **window** on the screen.
  - In standard Java, we must do this ourselves; it is called a `JFrame`.

- Sets up a **drawing canvas** in the center of the window
  In standard Java, we must create our own drawing canvas.

- Provides convenient methods to listen for mouse events.
  - In standard Java, event handling takes a bit more code to set up.

ACM GUI example

```java
public class ColorFun extends Program {
    public void init() {
        JButton button1 = new JButton("Red!");
        JButton button2 = new JButton("Blue!");
        add(button1, SOUTH);
        add(button2, SOUTH);
        addActionListeners();
    }
    public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand().equals("Red!")) {
            setBackground(Color.BLUE);
        } else {
            setBackground(Color.RED);
        }
    }
}
```

Java GUI example

```
public class ColorFun implements ActionListener {
    public static void main(String[] args) {
        new ColorFun().init();
    }
    private JFrame frame;
    public void init() {
        frame = new JFrame("ColorFun");
        frame.setSize(500, 300);
        JButton button1 = new JButton("Red!");
        JButton button2 = new JButton("Blue!");
        button1.addActionListener(this);
        button2.addActionListener(this);
        frame.add(button1, "South");
        frame.add(button2, "South");
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand().equals("Red!")) {
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
}   }   }
```

Summary

- **Benefits of libraries:**
  - simplify syntax/rough edges of language/API
  - avoid re-writing the same code over and over
  - possible to make advanced programs quickly
  - leverage work of others

Click to start new game

Lives: 3 Score: 0

- **Drawbacks of libraries:**
  - limitations on usage; e.g. ACM library cannot be re-distributed for commercial purposes

Programming Languages

Java
JavaScript
Python
Objective-C
Haskell
PHP
Pascal
Ruby
Perl
C++
C
Erlang
C#
D
Scala
F#
ML
OpenEdge ABL
ABAP
Visual Basic
Transact-SQL
Clarion
FoxPro
Logo
Prolog
ColdFusion
Forth
PL/SQL
PostScript
Q
Assembly
Scratch
COBOL
Scheme
R
Go
Fortran
ActionScript
MATLAB
Ada
Lua
Io
Yacc
TCL
Delphi
PL/I
Dart
Scala
Max/MSP
cT
SAS
Haskell
Tcl
Groovy
Lisp
Mathematica
Swift

Java

```java
ArrayList<Double> evens = new ArrayList<>();
for(int i = 0; i < 100; i++) {
    if(i % 2 == 0) {
        evens.add(i);
    }
}
println(evens);
```

prints [2, 4, 6, 8, 10, 12, …]

C++

```
Vector<double> evens;
for(int i = 0; i < 100; i++) {
    if(i % 2 == 0) {
        evens.add(i);
    }
}
cout << evens << endl;
```

```
prints [2, 4, 6, 8, 10, 12, … ]
```

Python

```
evens = []
for i in range(100):
    if i % 2 == 0:
        evens.append(i)
print evens
```

prints [2, 4, 6, 8, 10, 12, …]

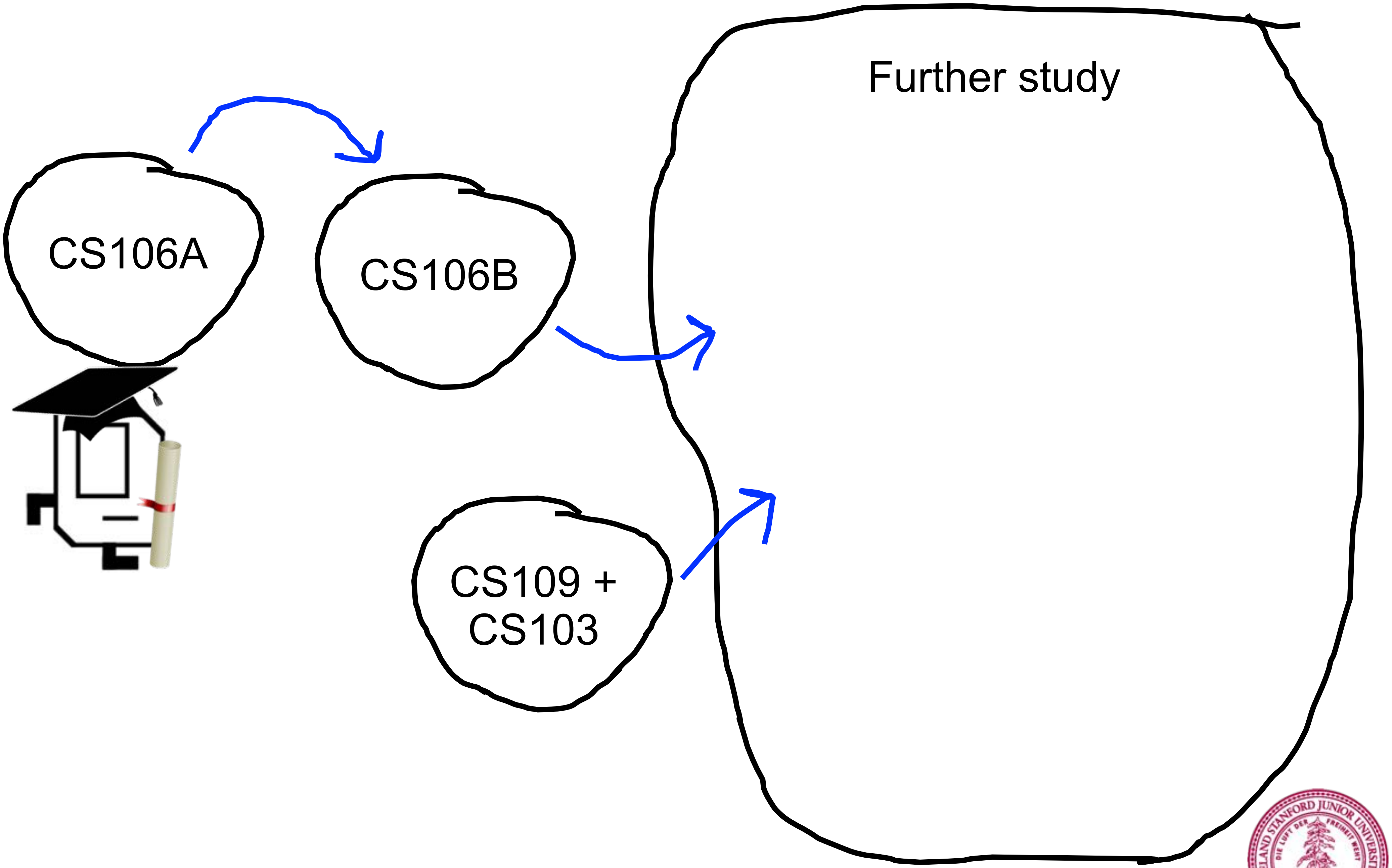
Javascript

```
var evens = []
for(var i = 0; i < 100; i++) {
    if(i % 2 == 0) {
        evens.push(i)
    }
}
console.log(evens)
```

```
prints [2, 4, 6, 8, 10, 12, … ]
```

Future in the CS curriculum

The CS Curriculum



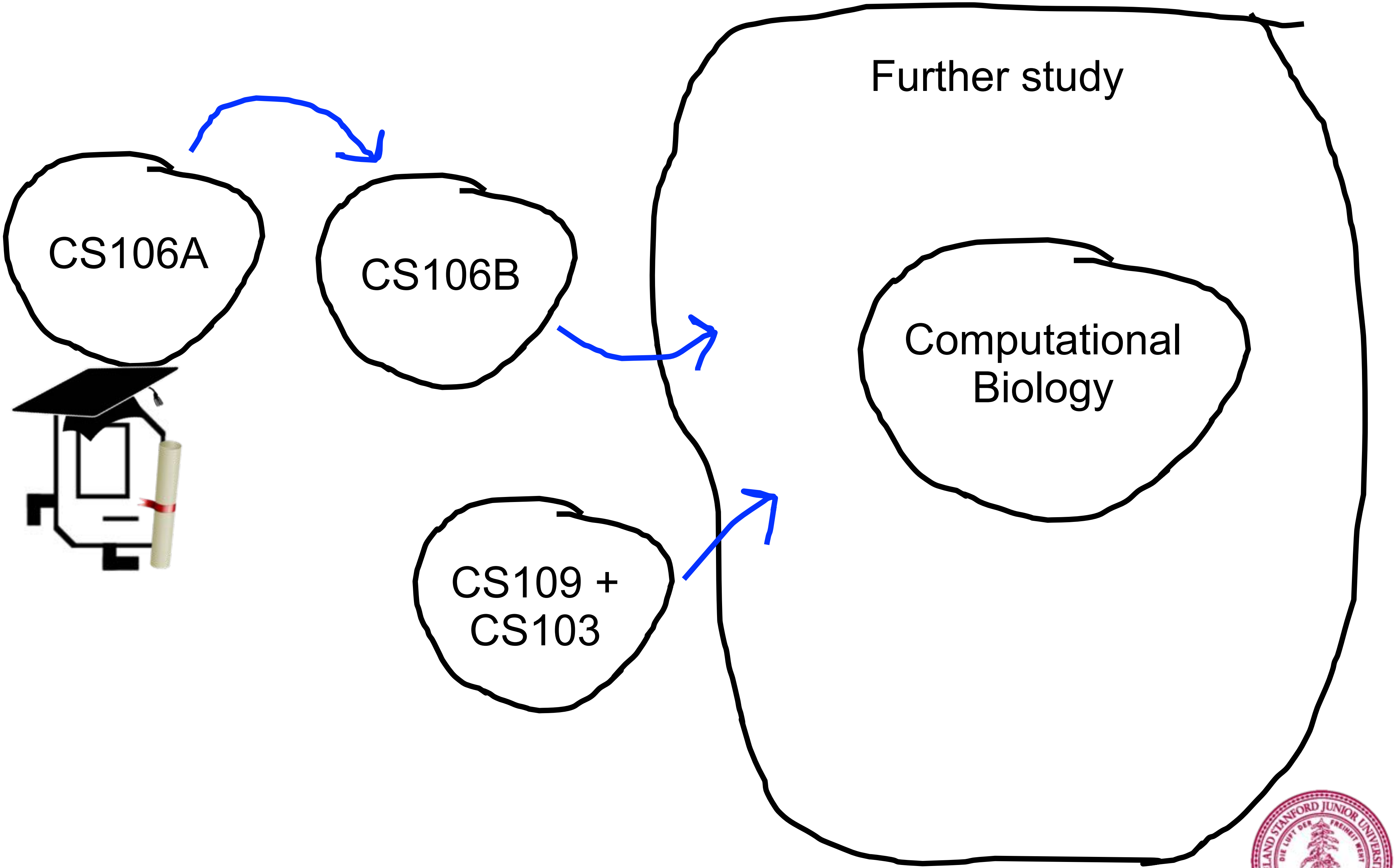
CS106A

CS106B

CS109 + CS103

Further study

The CS Curriculum



CS106A

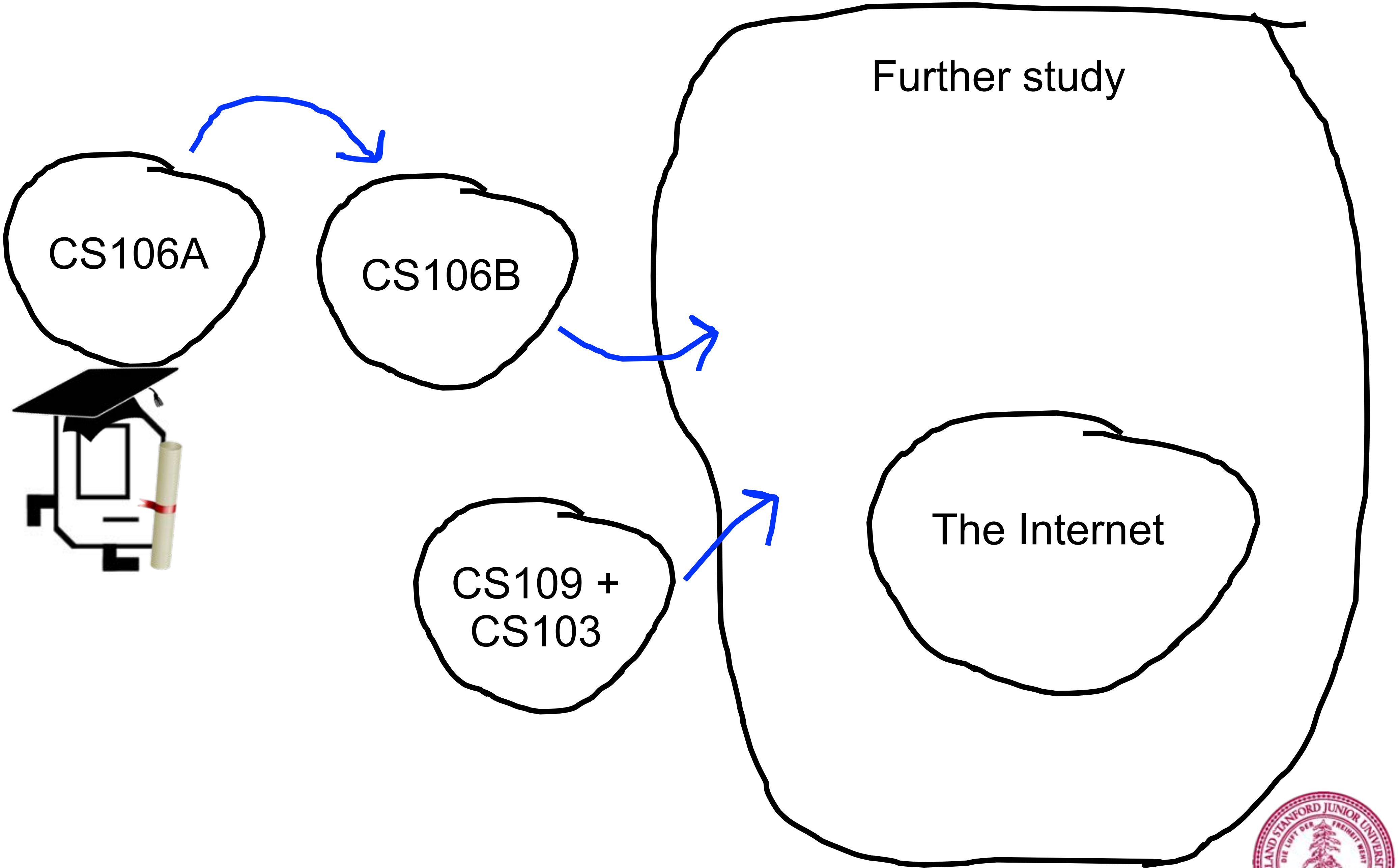
CS106B

CS109 + CS103

Further study

Computational Biology

The CS Curriculum



CS106A

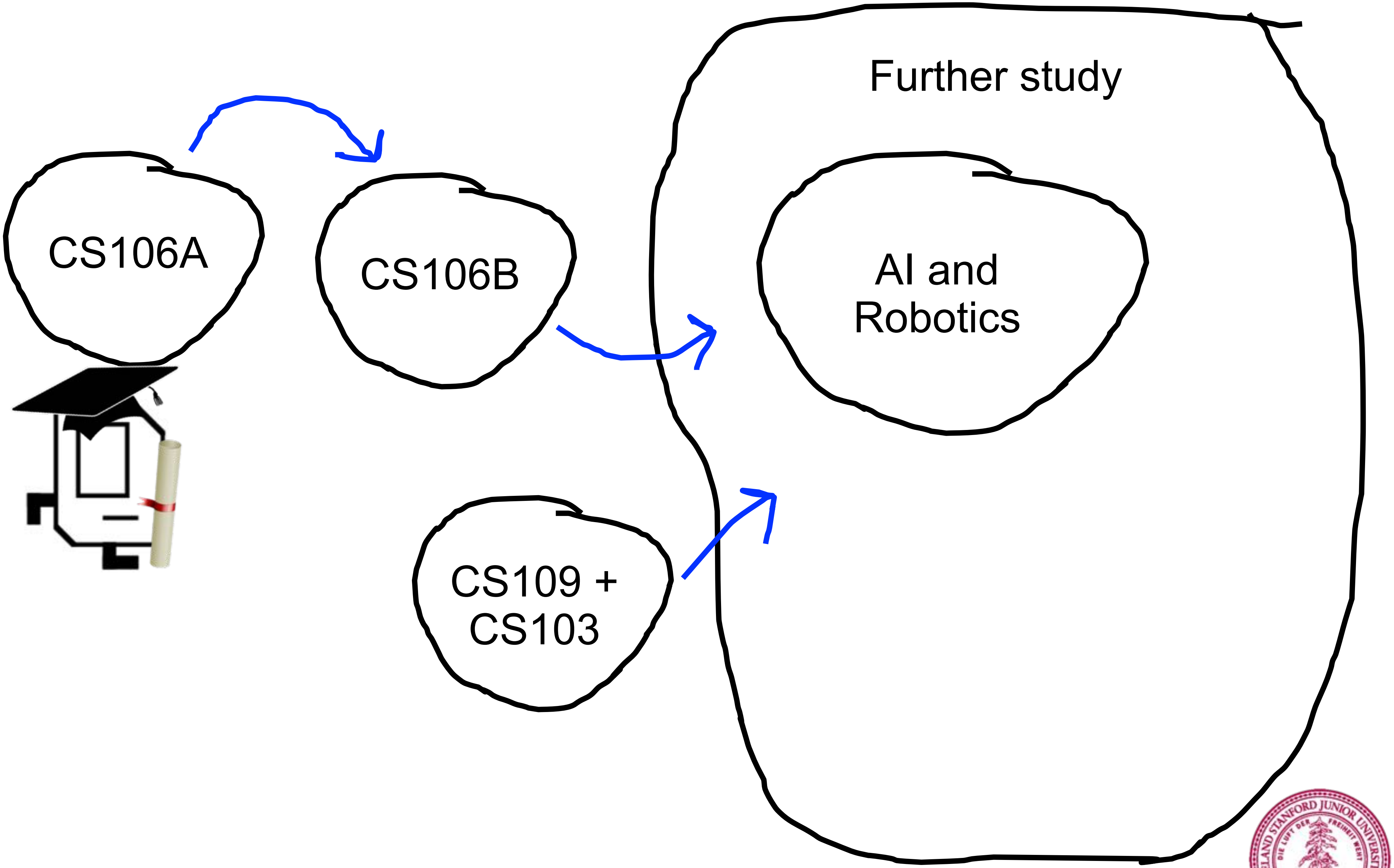
CS106B

CS109 + CS103

Further study

The Internet

The CS Curriculum



CS106A

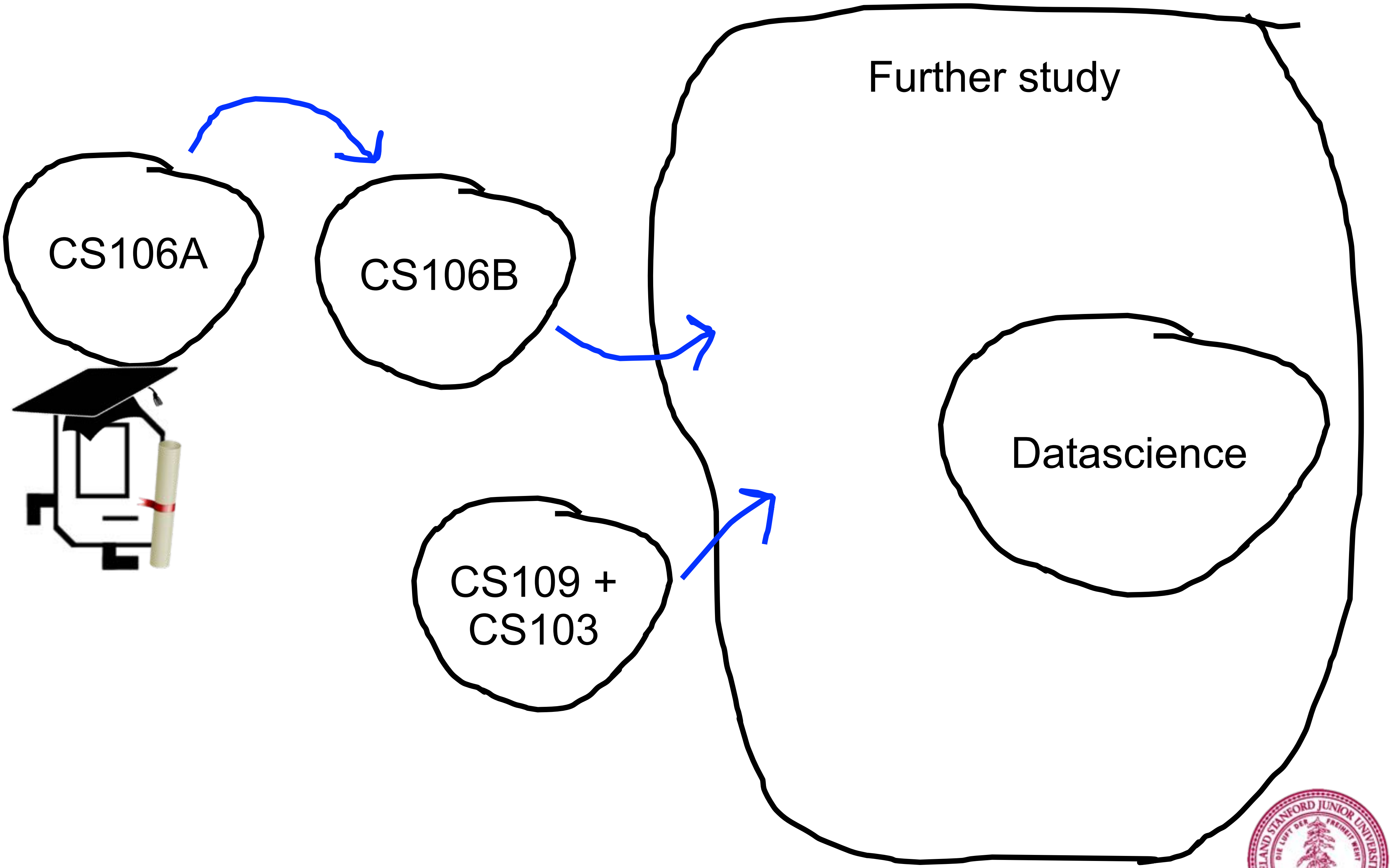
CS106B

CS109 + CS103

Further study

AI and Robotics

The CS Curriculum



CS106A

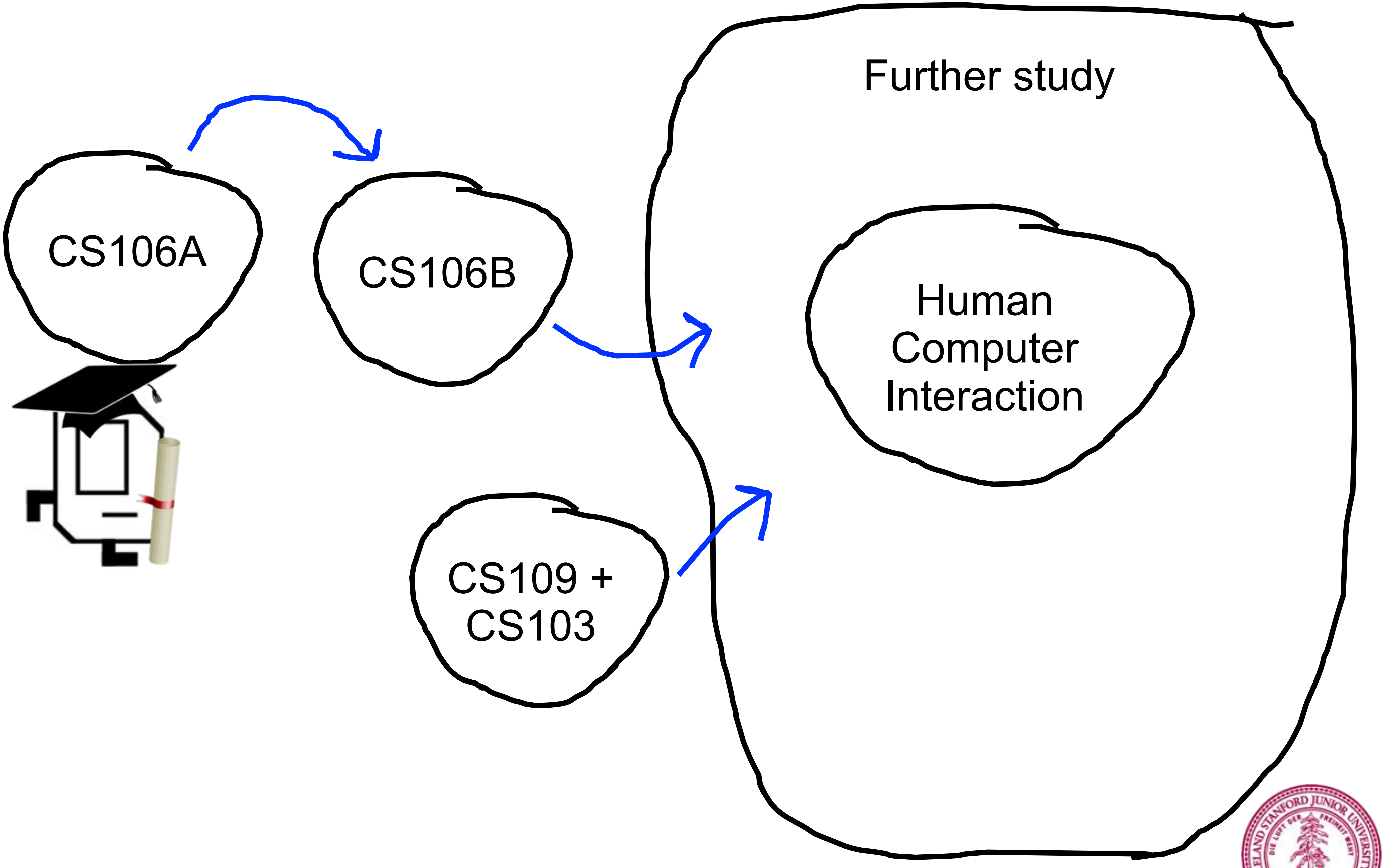
CS106B

CS109 + CS103

Further study

Datascience

The CS Curriculum



CS106A

CS106B

CS109 + CS103

Further study

Human Computer Interaction

Machine Learning

Machine Learning
or, How we learned to decompose

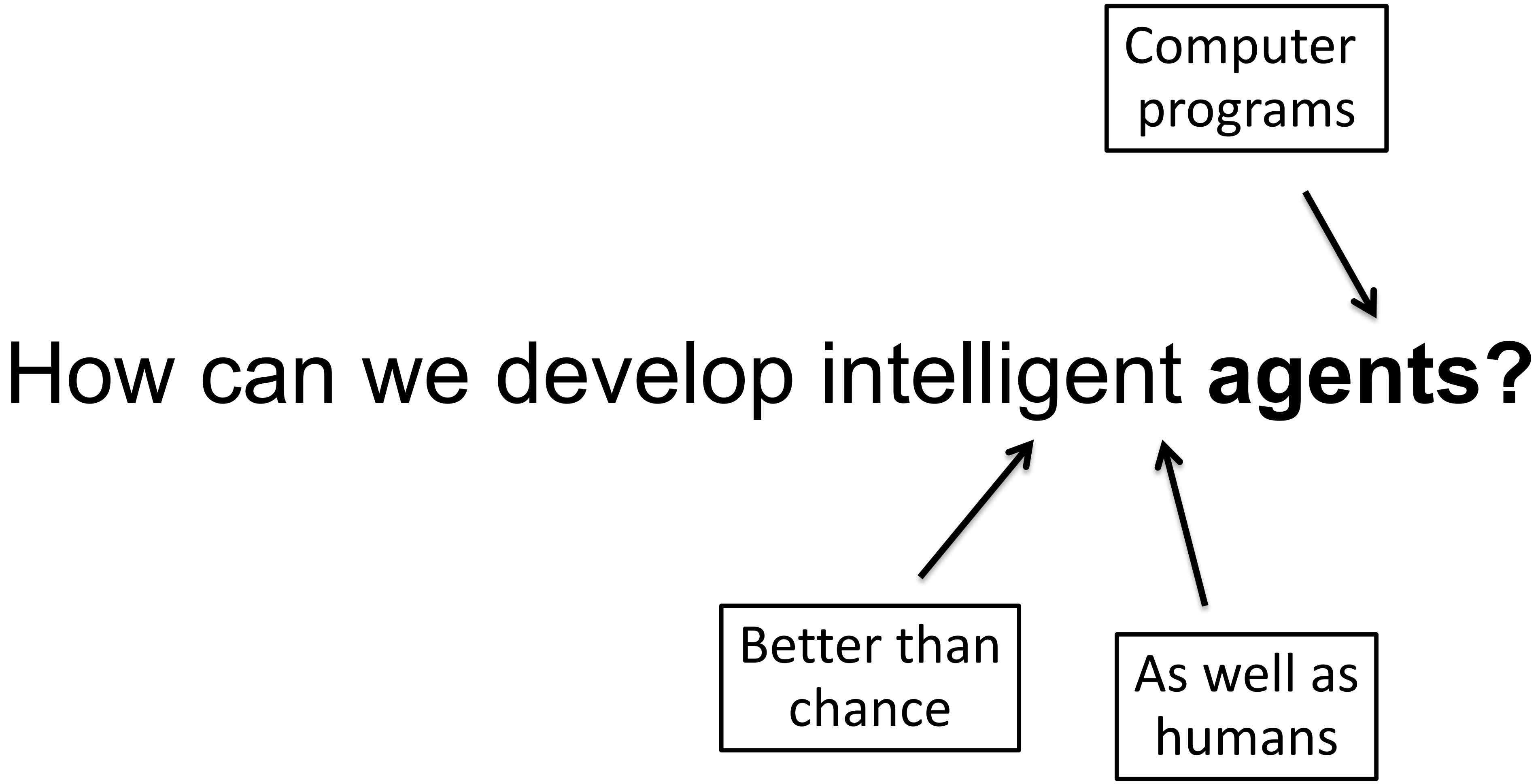
There is something going on
in the world of AI

Something big (for us)…

[suspense]

How can we develop intelligent **agents?**

Volunteer



Computer programs

How can we develop intelligent **agents?**

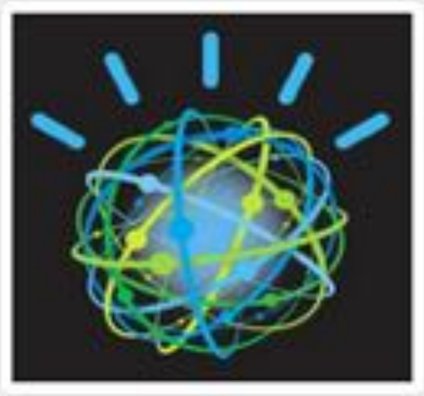
Better than chance

As well as humans

Big Milestones Starting in 1997

1997 Deep Blue

2005 Stanley



2011 Watson

Self Driving Cars

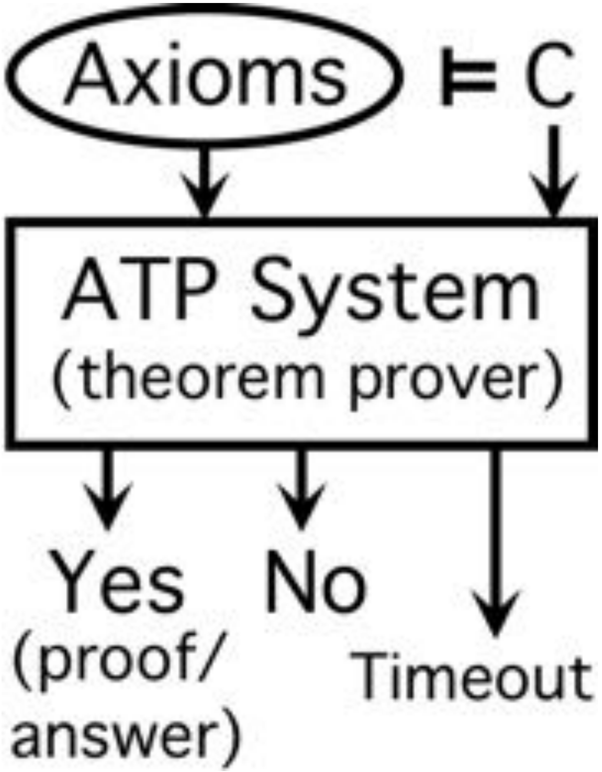
Computers Making Art

The Last Remaining Board Game

Early Optimism 1950

1952

1955

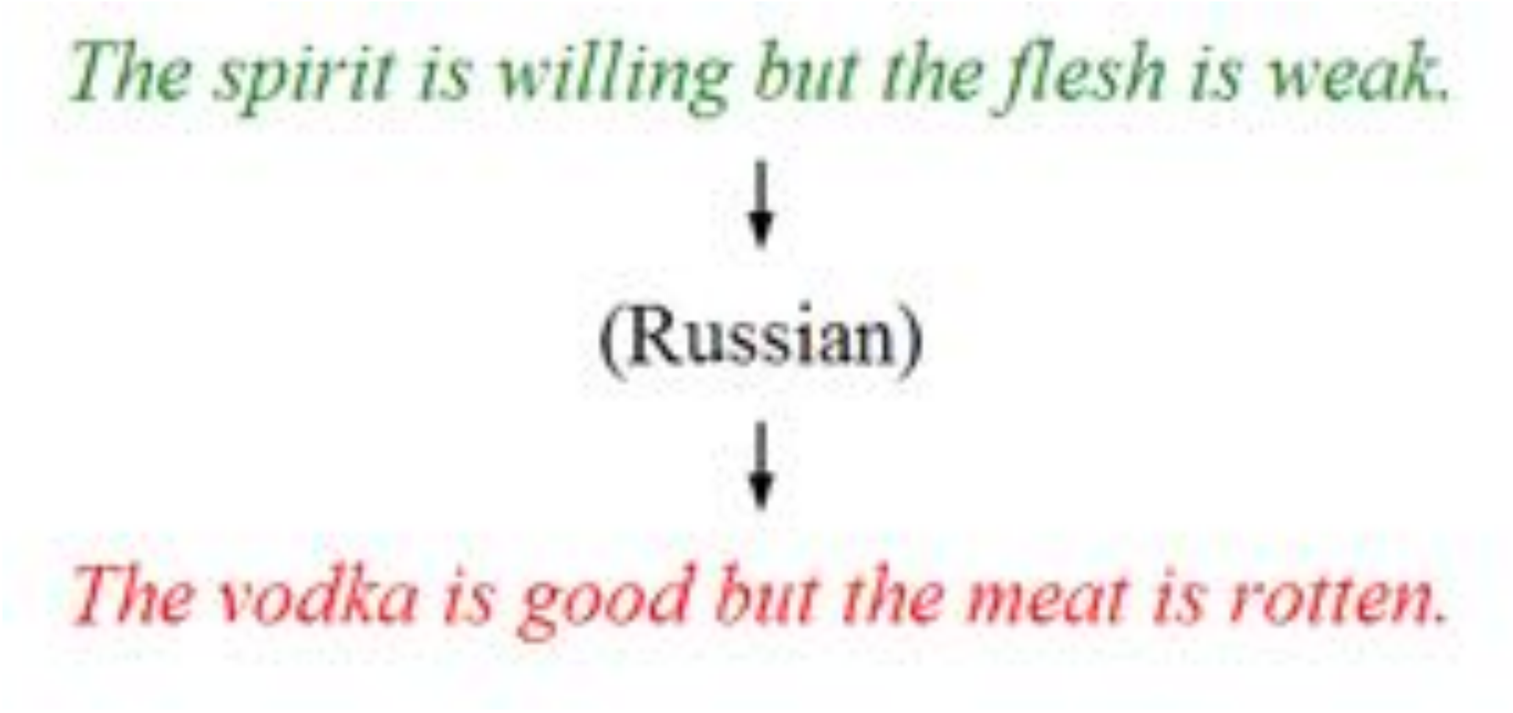


Early Optimism 1950

"Machines will be capable, within twenty years, of doing any work a man can do."

–Herbert Simon, 1952

Underwhelming Results 1950s to 1980s



The spirit is willing but the flesh is weak.

↓

(Russian)

↓

The vodka is good but the meat is rotten.

The world is too complex

BRACE YOURSELVES
WINTER IS COMING

Simple Example: Identifying Cats

- We have a picture and we want to know if it’s a cat or not.

→ `true`

→ `false`

→ `true`

→ `false`

F. Deckel-München
kirkh on deviantart
COMPUR
2.8
4
5.6
8
11
16
22

Identifying Cats

Here’s one way you might code this…

```
private void isCat(GImage animal) {
   int[][] pixels = animal.getPixelArray();
   if (containsTwoEyes(pixels)){
      if (hasWhiskers(pixels)){
         if (hasPointyEars(pixels)){
            return true;
         }
      }
   }
   return false;
}
```

Some Tricky Cases

Pros / Cons

- Pros
  - Matches our human intuition about what a cat is
  - Easy to understand the code
- Cons
  - Requires us to explicitly enumerate every feature that's important, and know how important it is
  - Need to write code to detect eyes, and whiskers, and the pointiness of ears
  - Will never improve… cannot learn from its mistakes

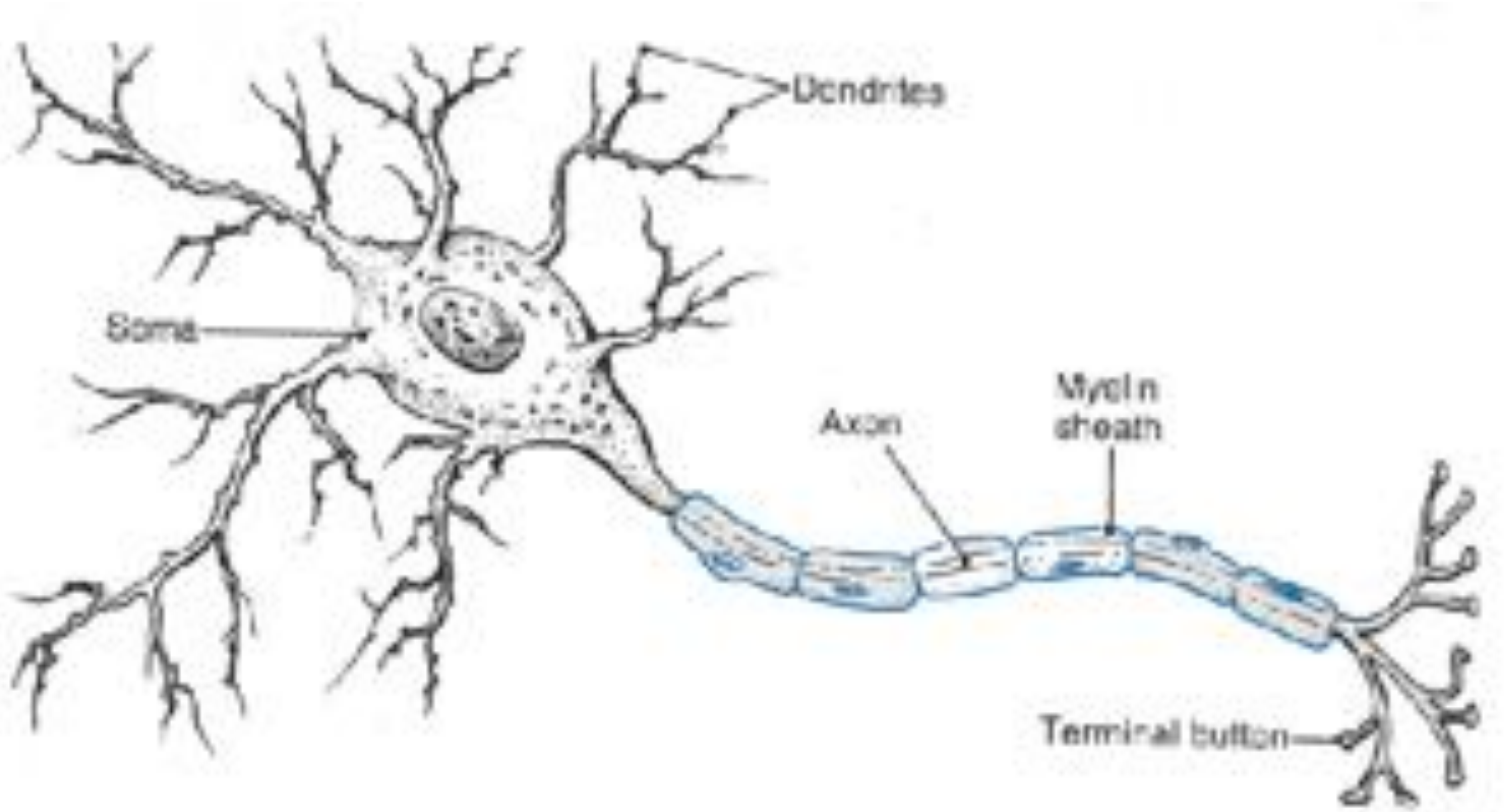
Hard problems seemed impossible.

Great idea #1 learn from experience

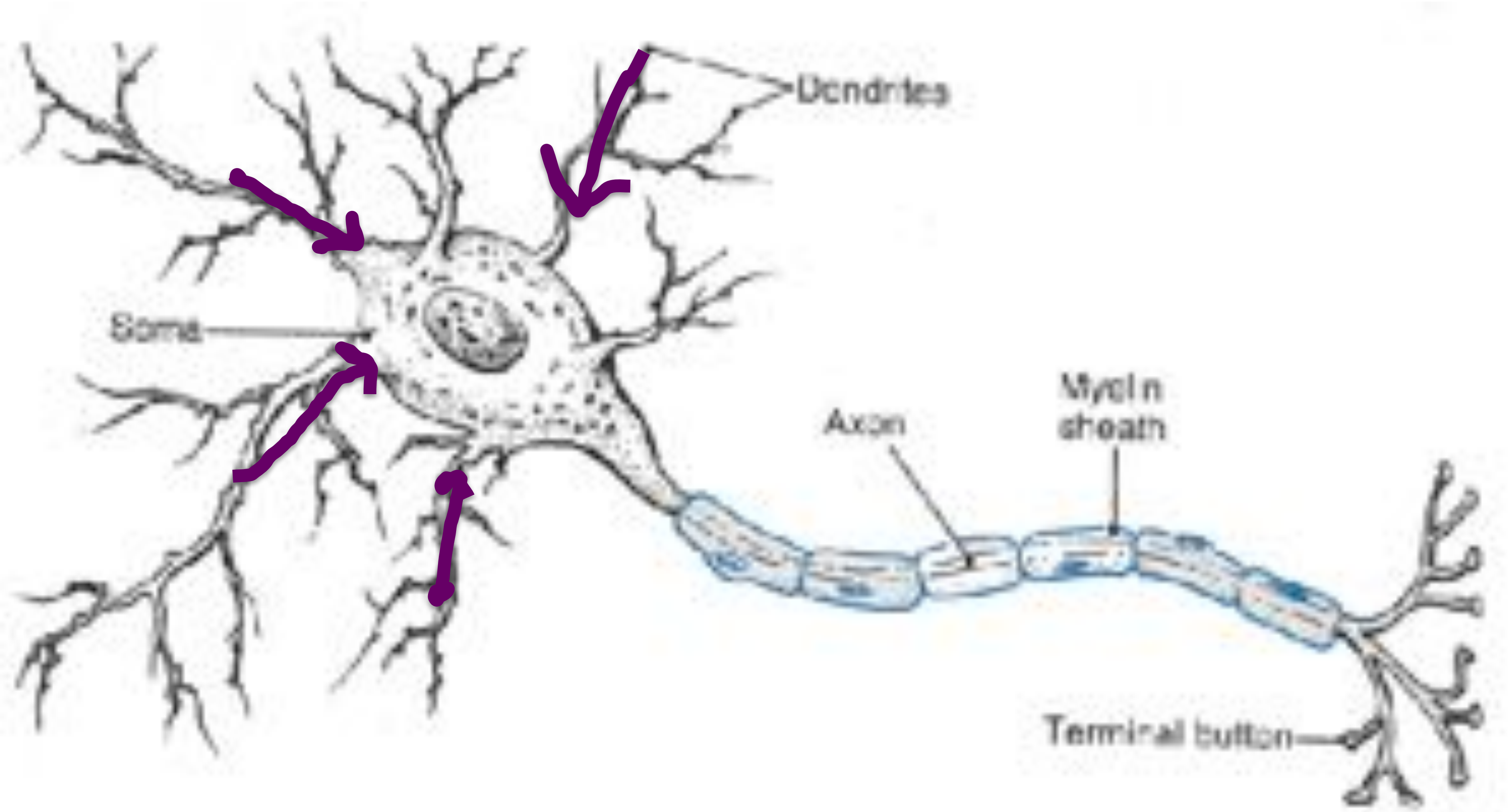
Machine Learning: Learn From Experience

Great idea #2 inspired by biology

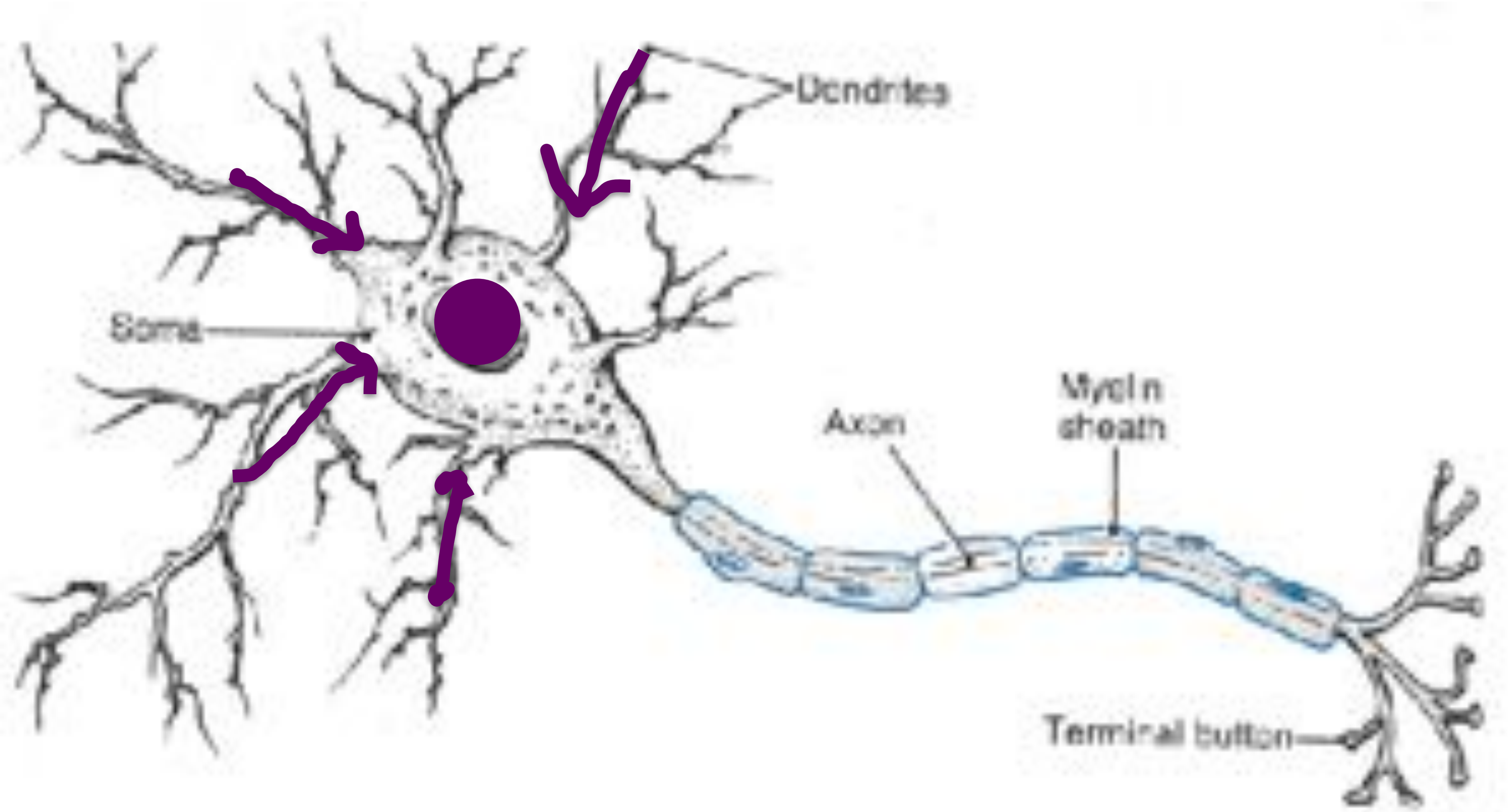
Neuron



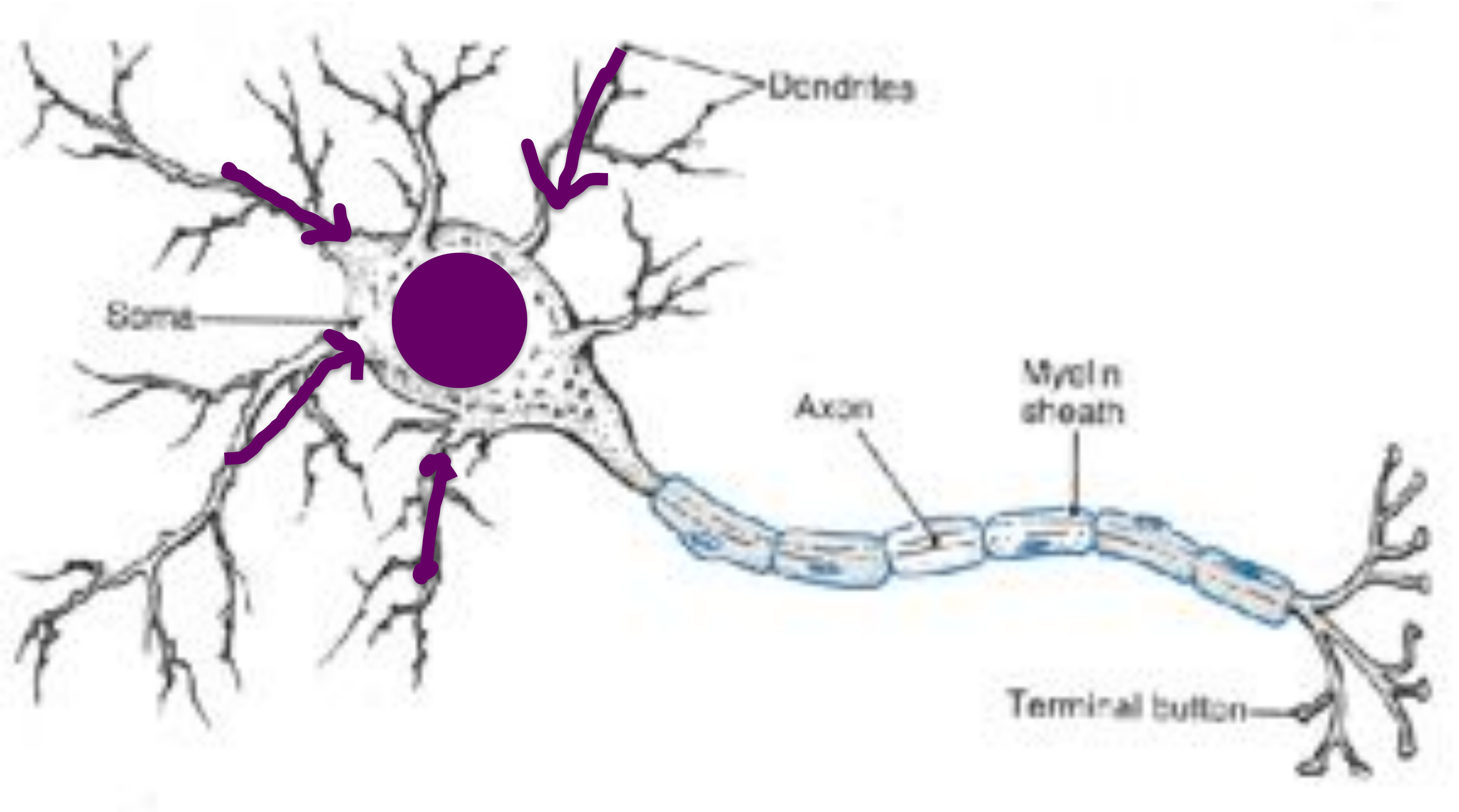
Neuron



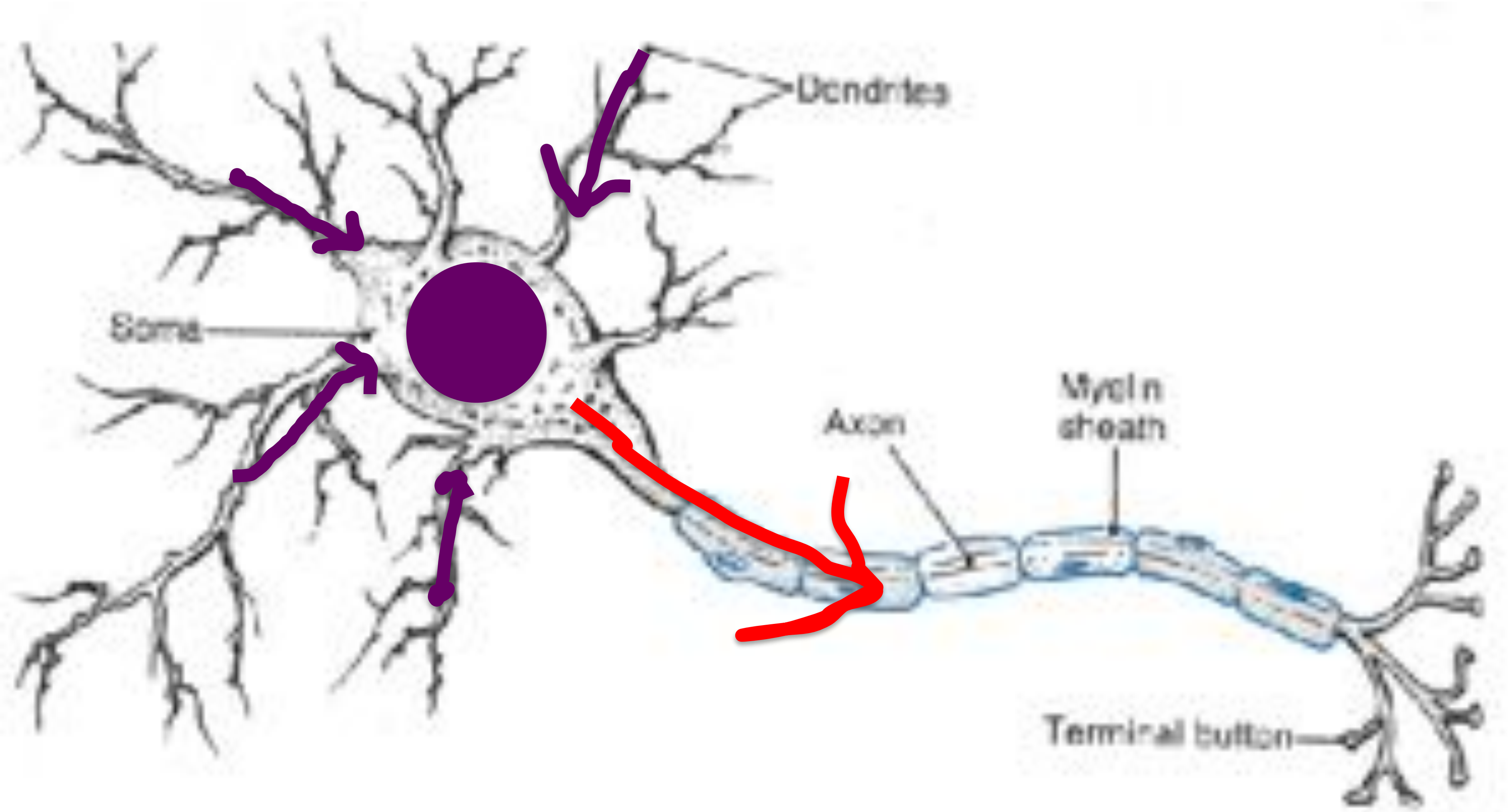
Neuron



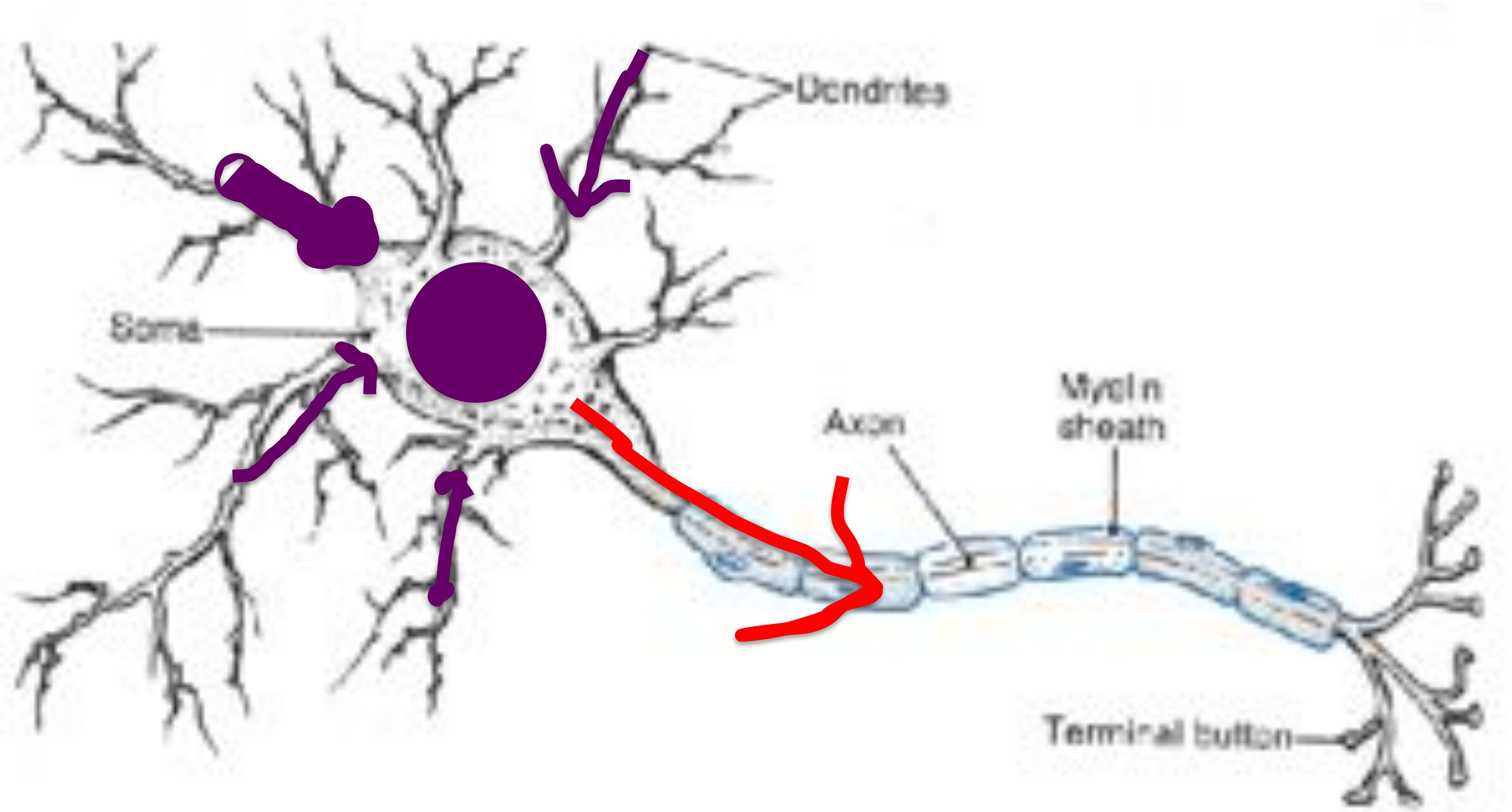
Neuron



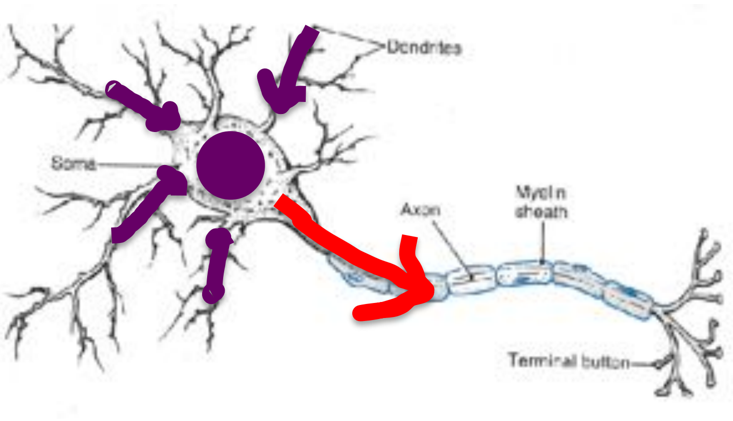
Neuron

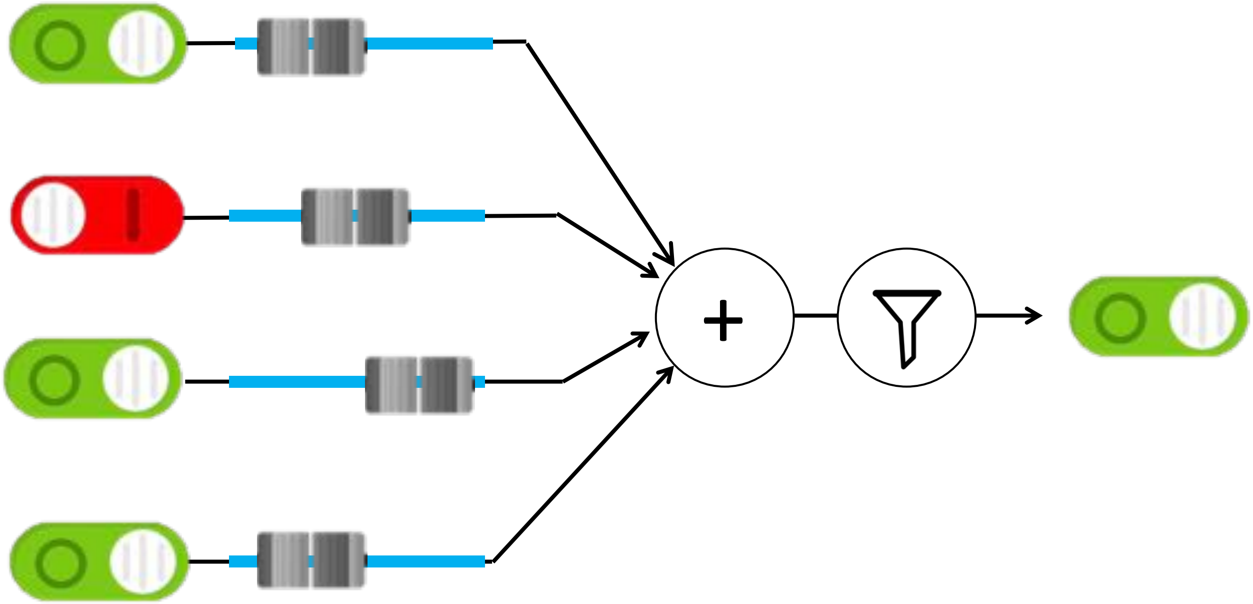


Some Inputs are More Important

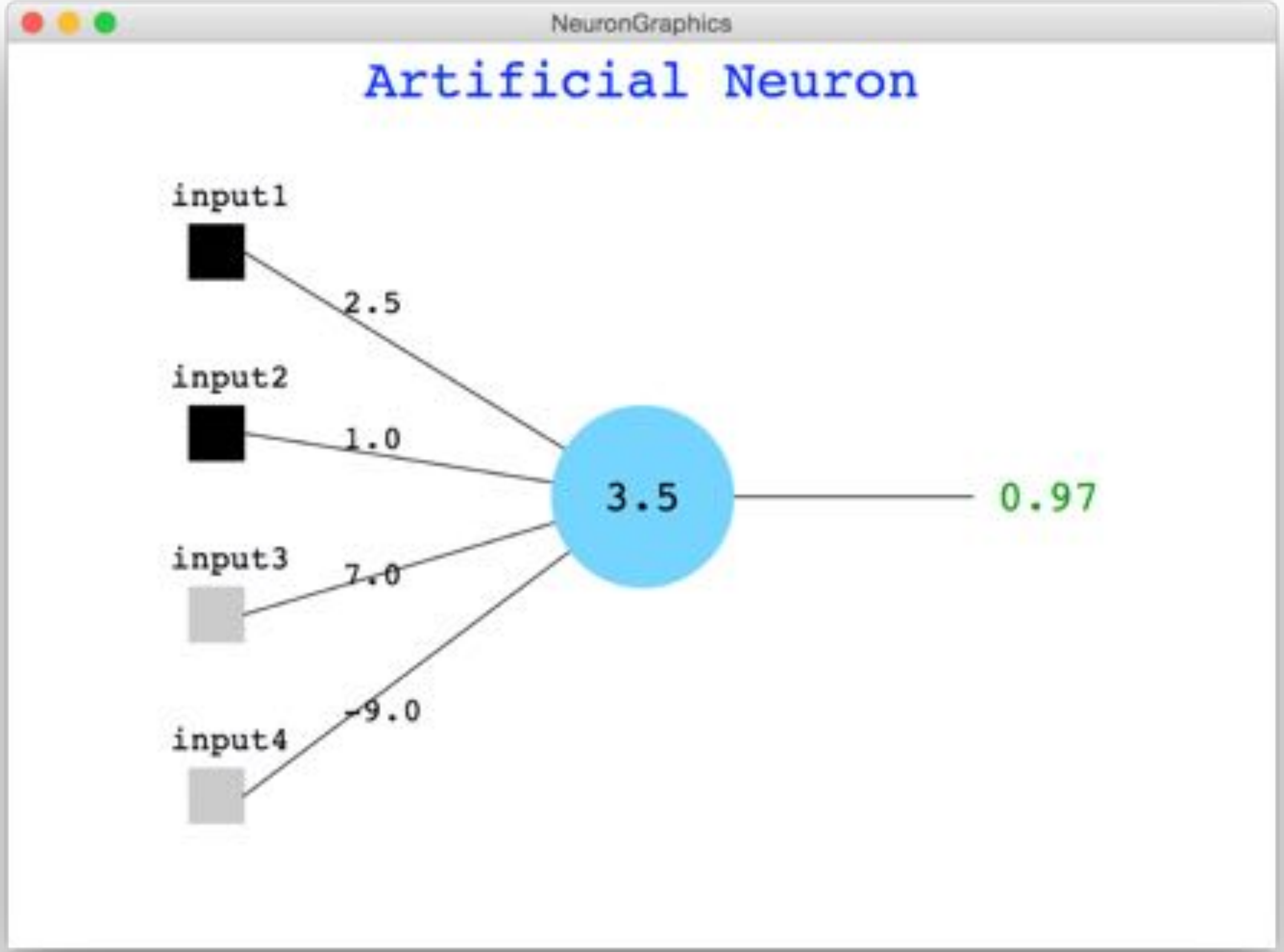


Artificial Neurons



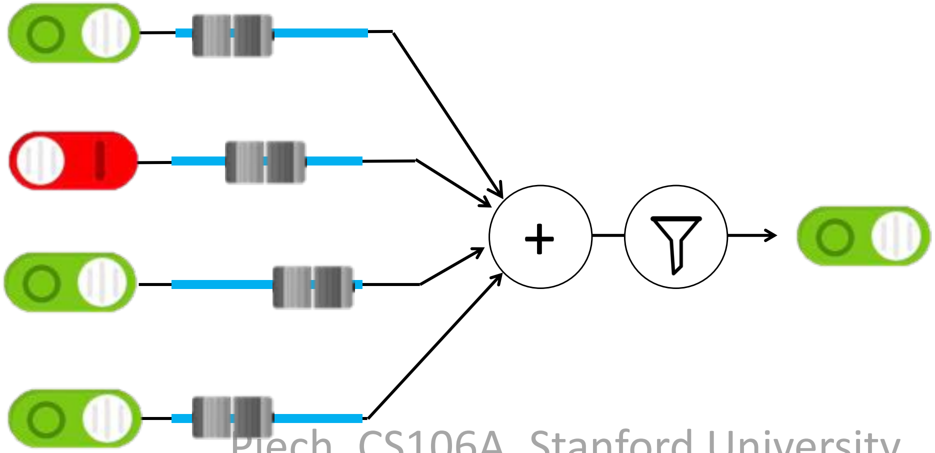


Java Demo


NeuronGraphics
Artificial Neuron
input1
input2
input3
input4
2.5
1.0
7.0
-9.0
3.5
0.97

Artificial Neuron

```
// calculate the activation of a neuron
private double activate(double[] weights, double[] inputs) {
    double weightedSum = 0;
    for(int i = 0; i < inputs.length; i++) {
        weightedSum += weights[i] * inputs[i];
    }

    return squash(weightedSum);

}


// the sigmoid function forces a value to be between 0 and 1
private double squash(double value) {
    return 1 / (1 + Math.exp(-value));
}
```



Digit Recognition Example

Let’s make feature vectors from pictures of numbers

input $= [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0]$

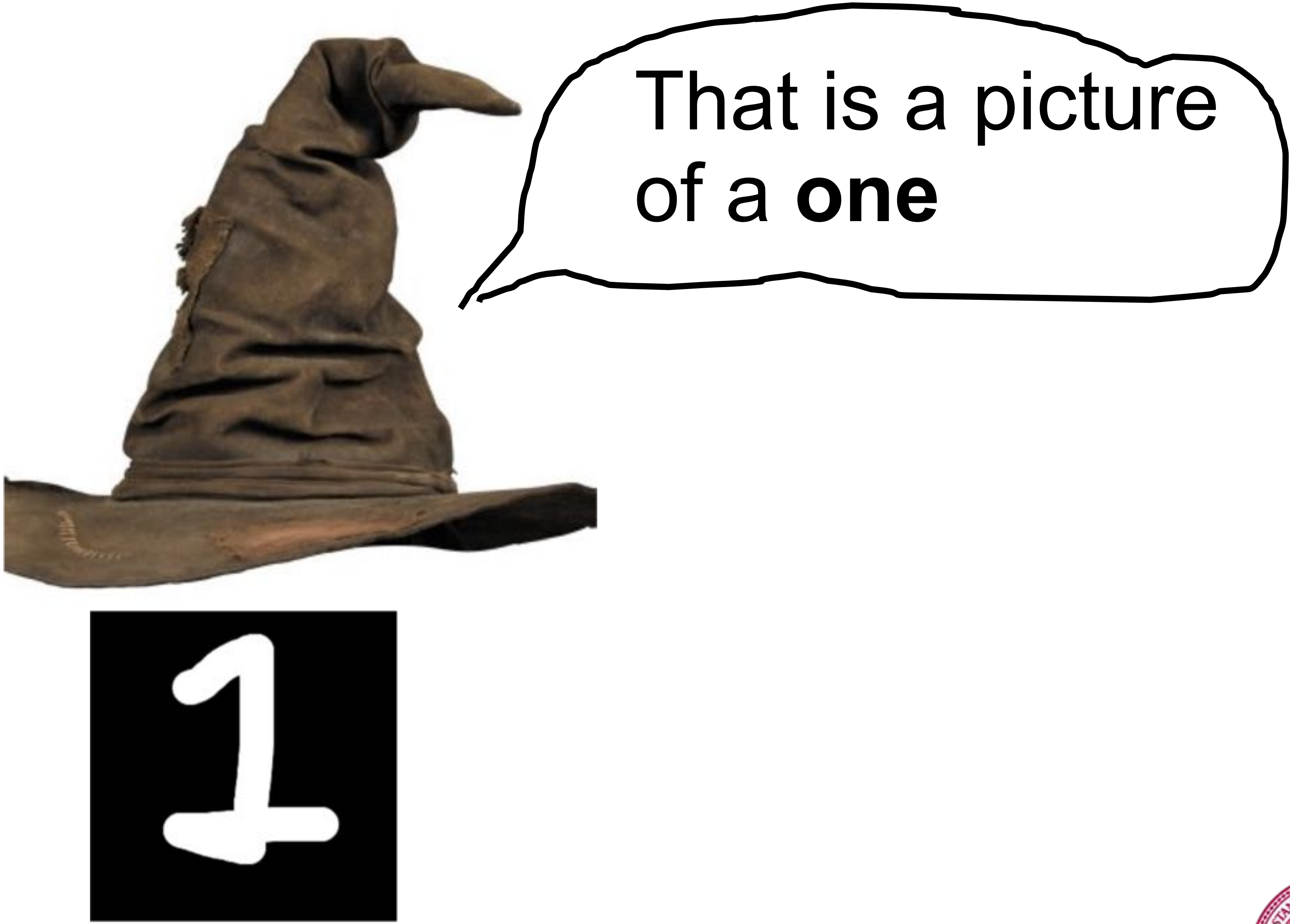
label $= 0$

input $= [0, 0, 1, 1, \ldots, 0, 1, 1, 0, \ldots 0, 1, 0, 0]$

label $= 1$

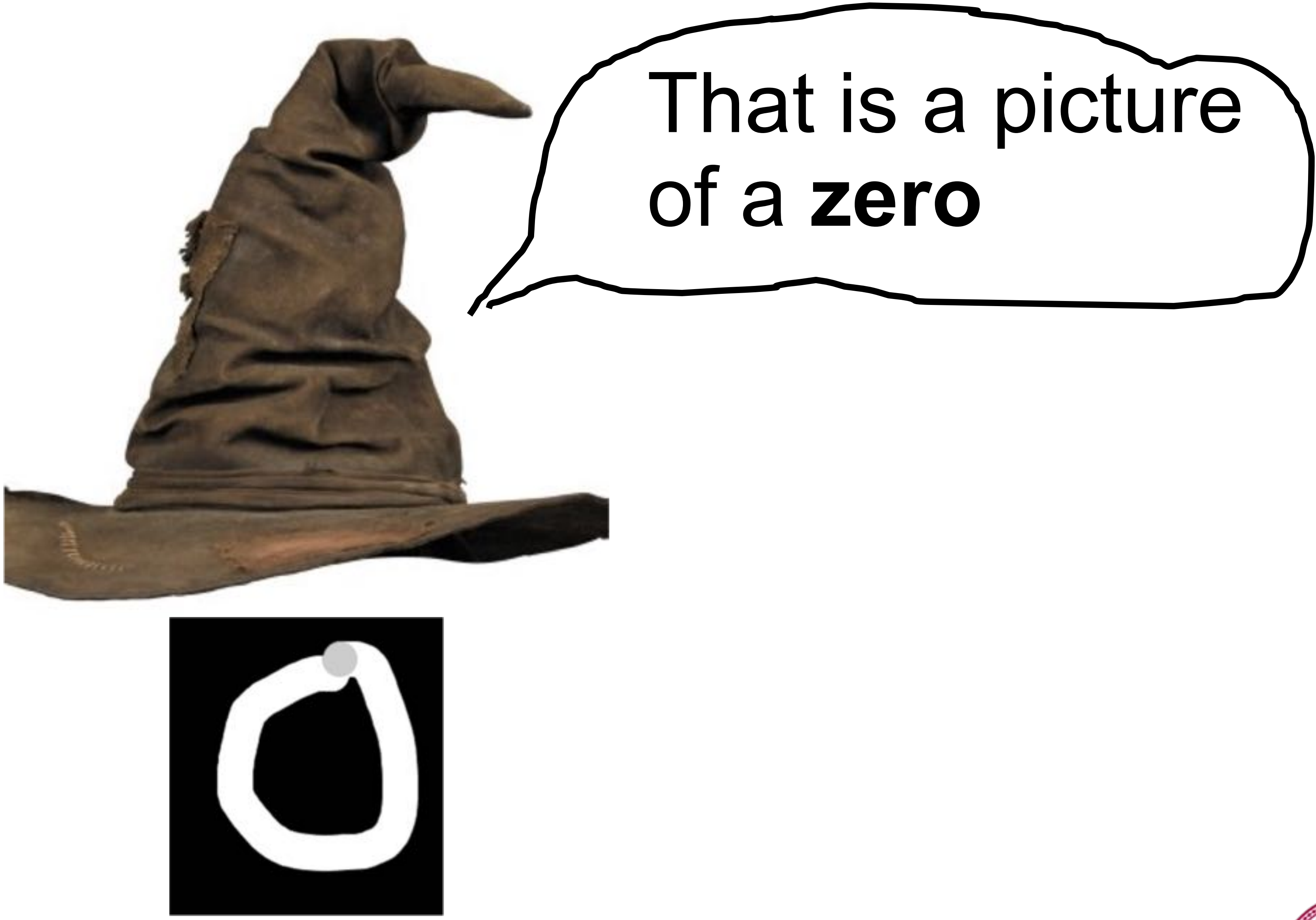
Computer Vision

Classification



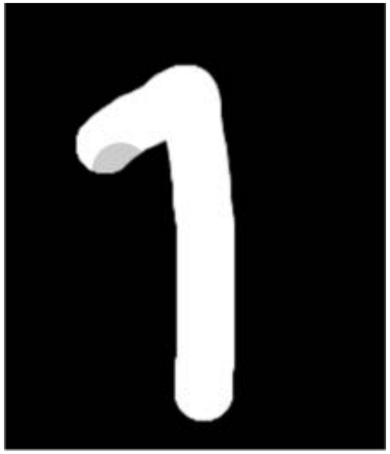
That is a picture of a **one**

Classification



Classification

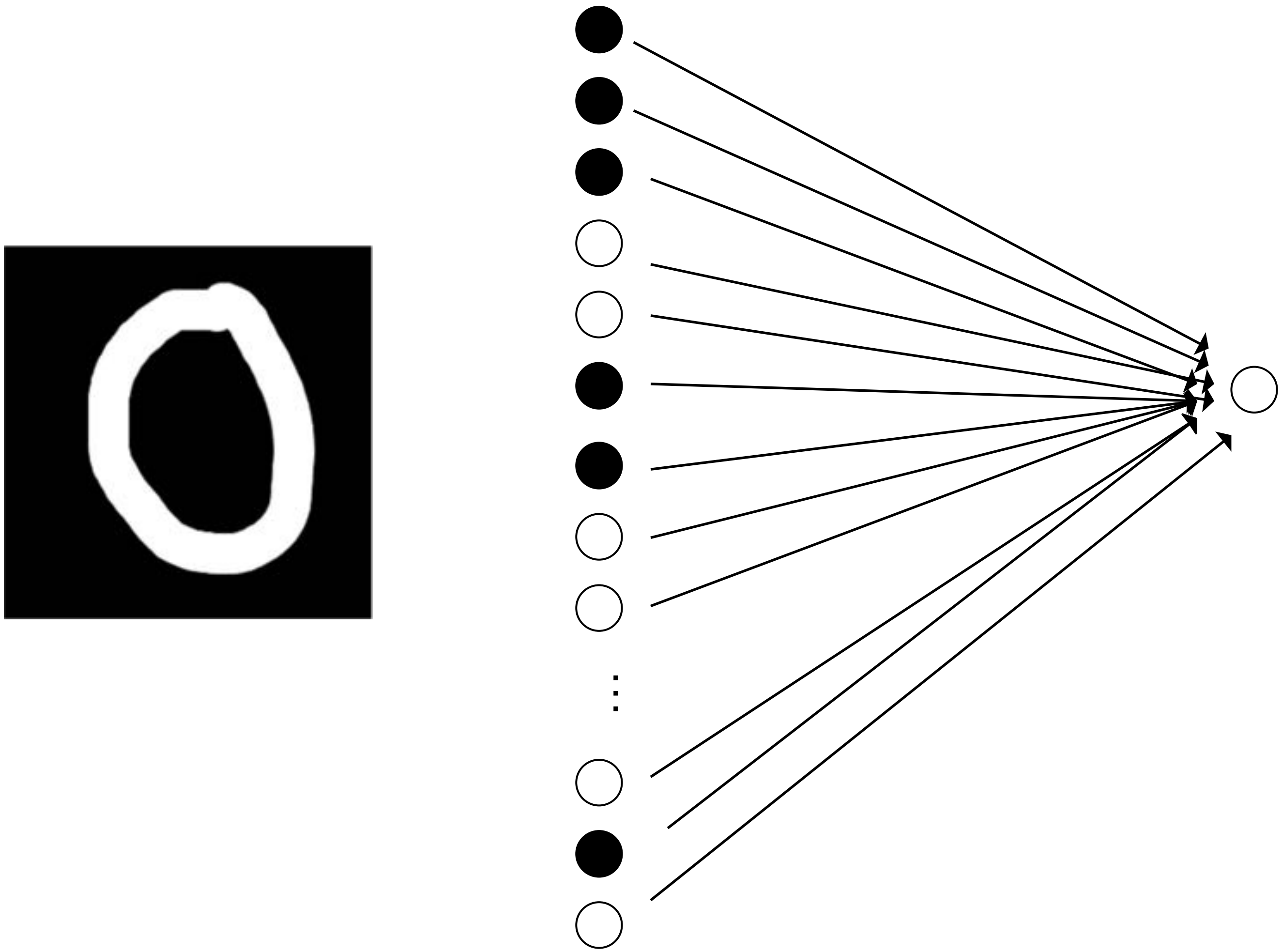
That is a picture of an **zero**



* It doesn’t have to be correct all of the time

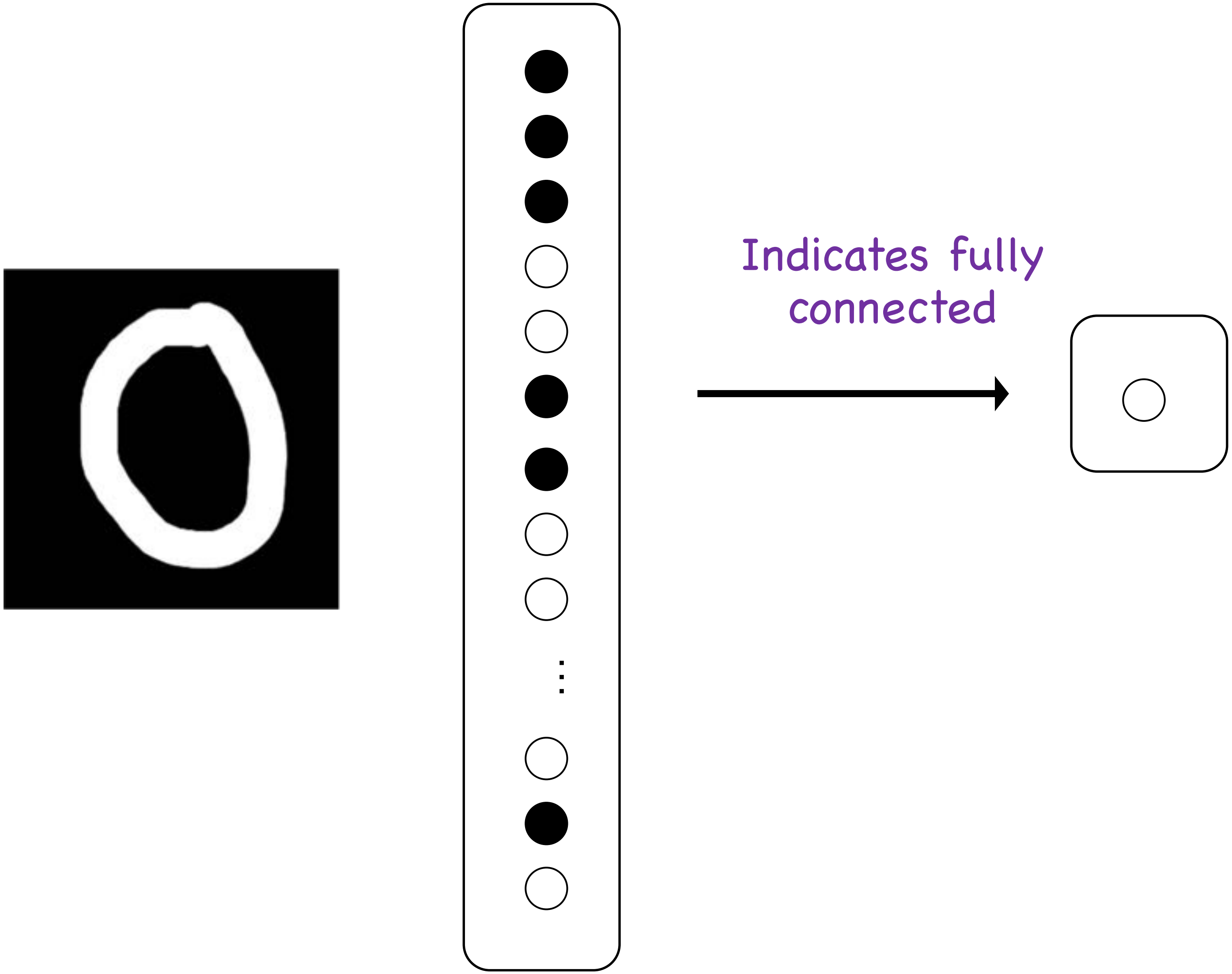
Can you do it?

Single Neuron



This means it predicts a 0

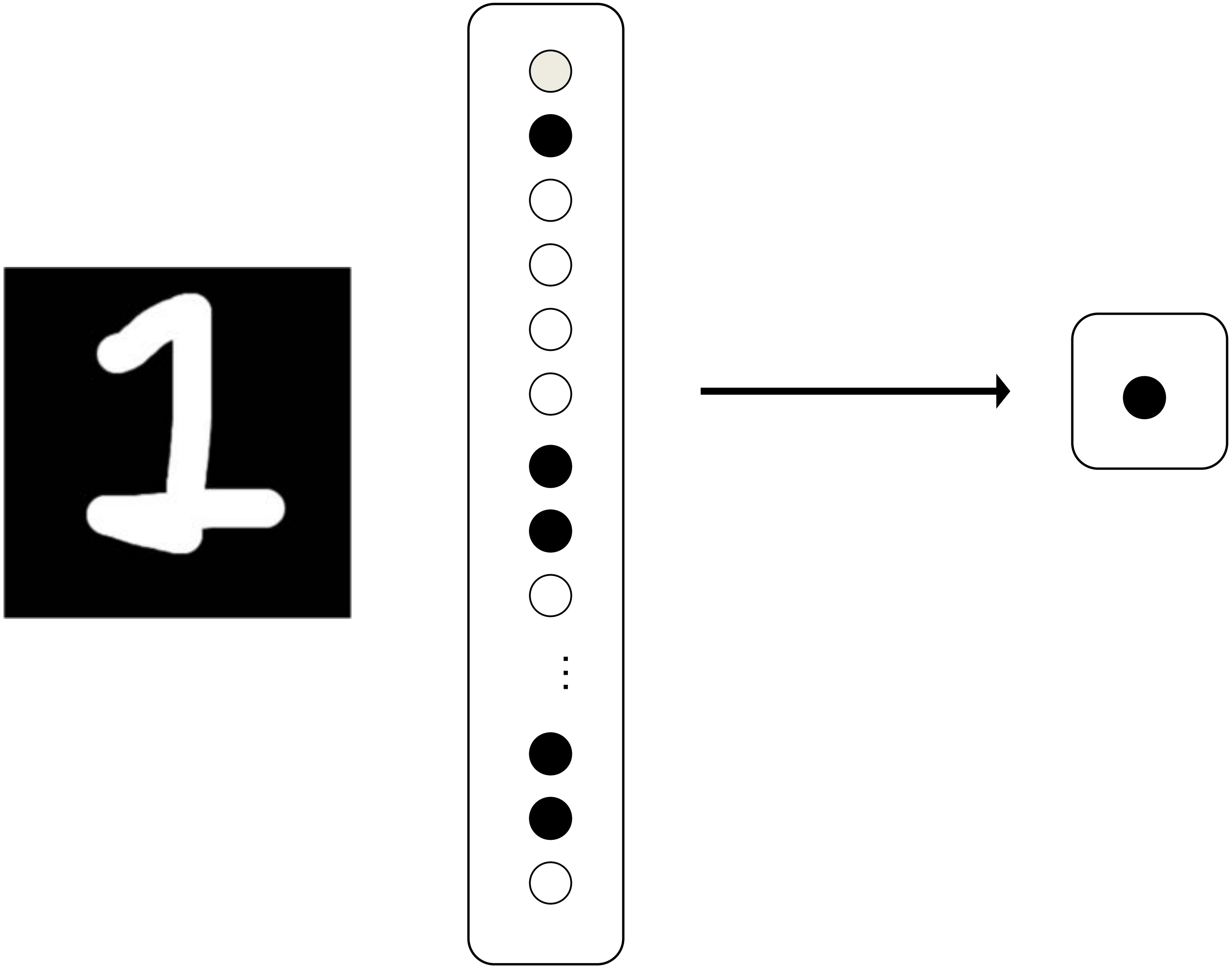
Single Neuron



Indicates fully connected

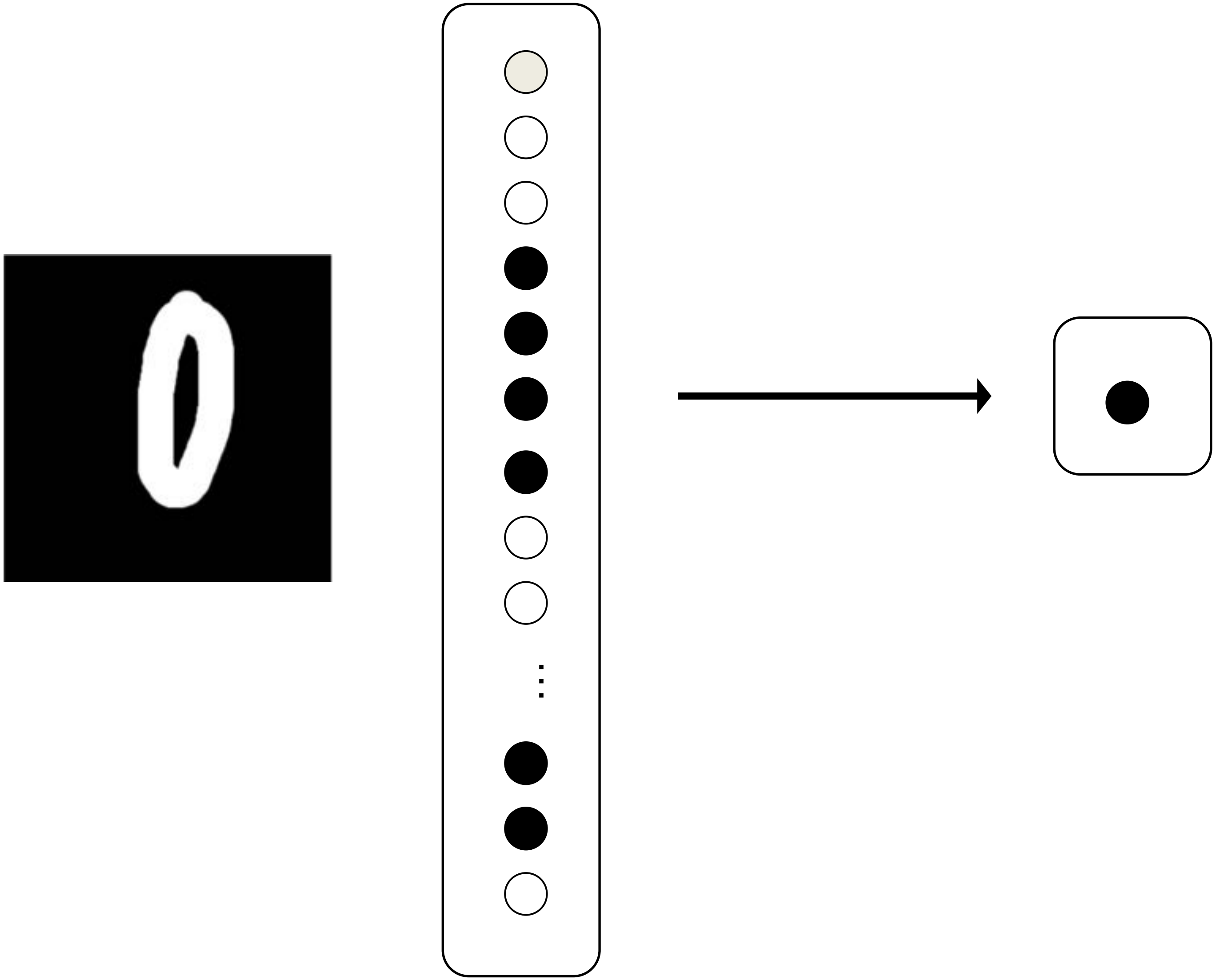
This means it predicts a 0

Single Neuron



This means it predicts a 1

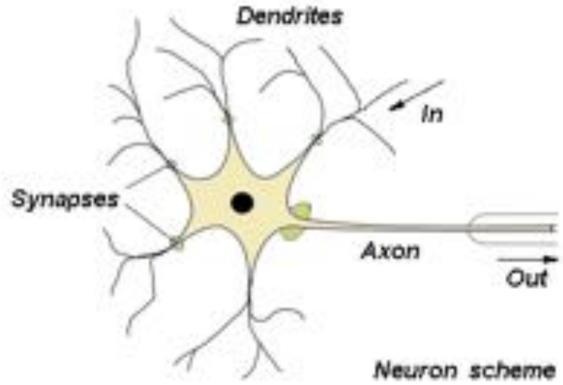
Not So Good

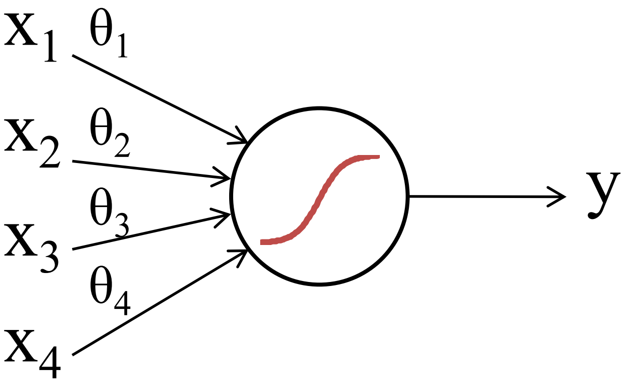


This means it predicts a 1

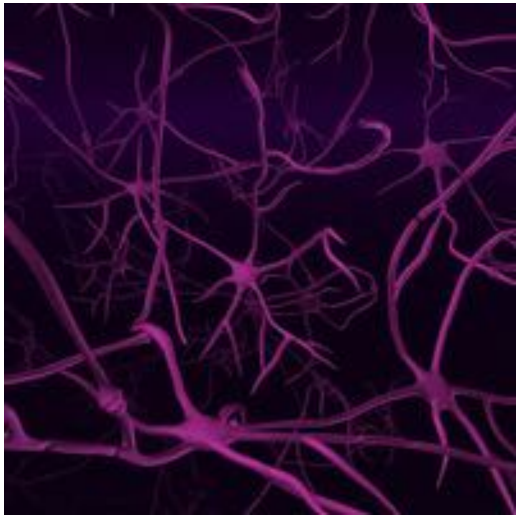
Biological Basis for Neural Networks

- A neuron

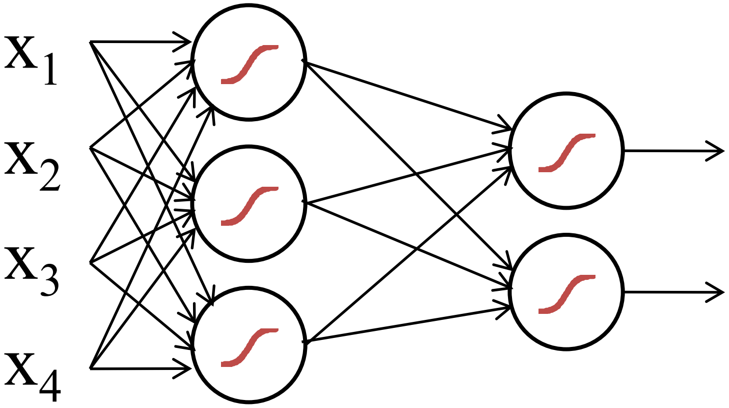




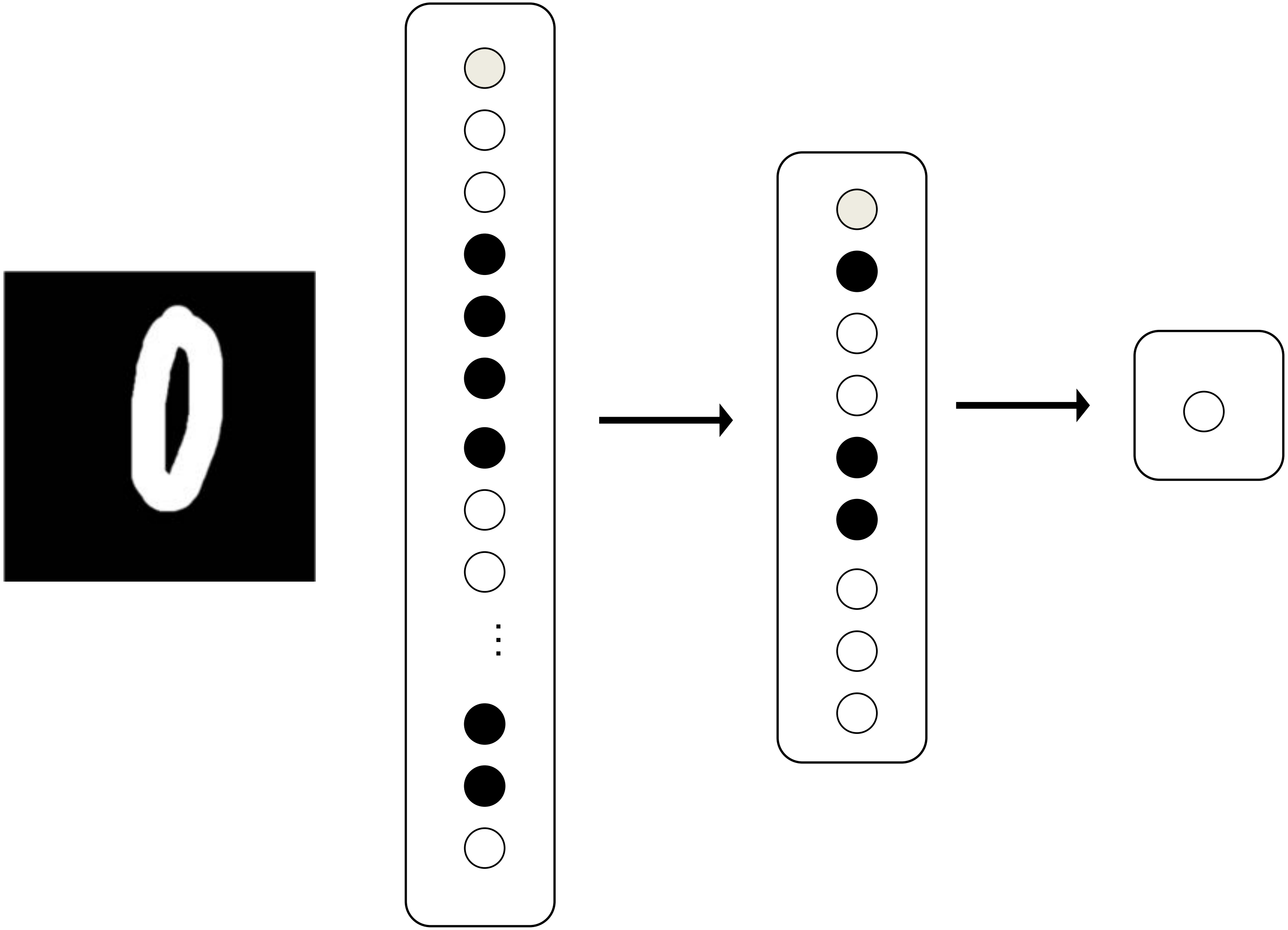
- Your brain



Actually, it's probably someone else's brain

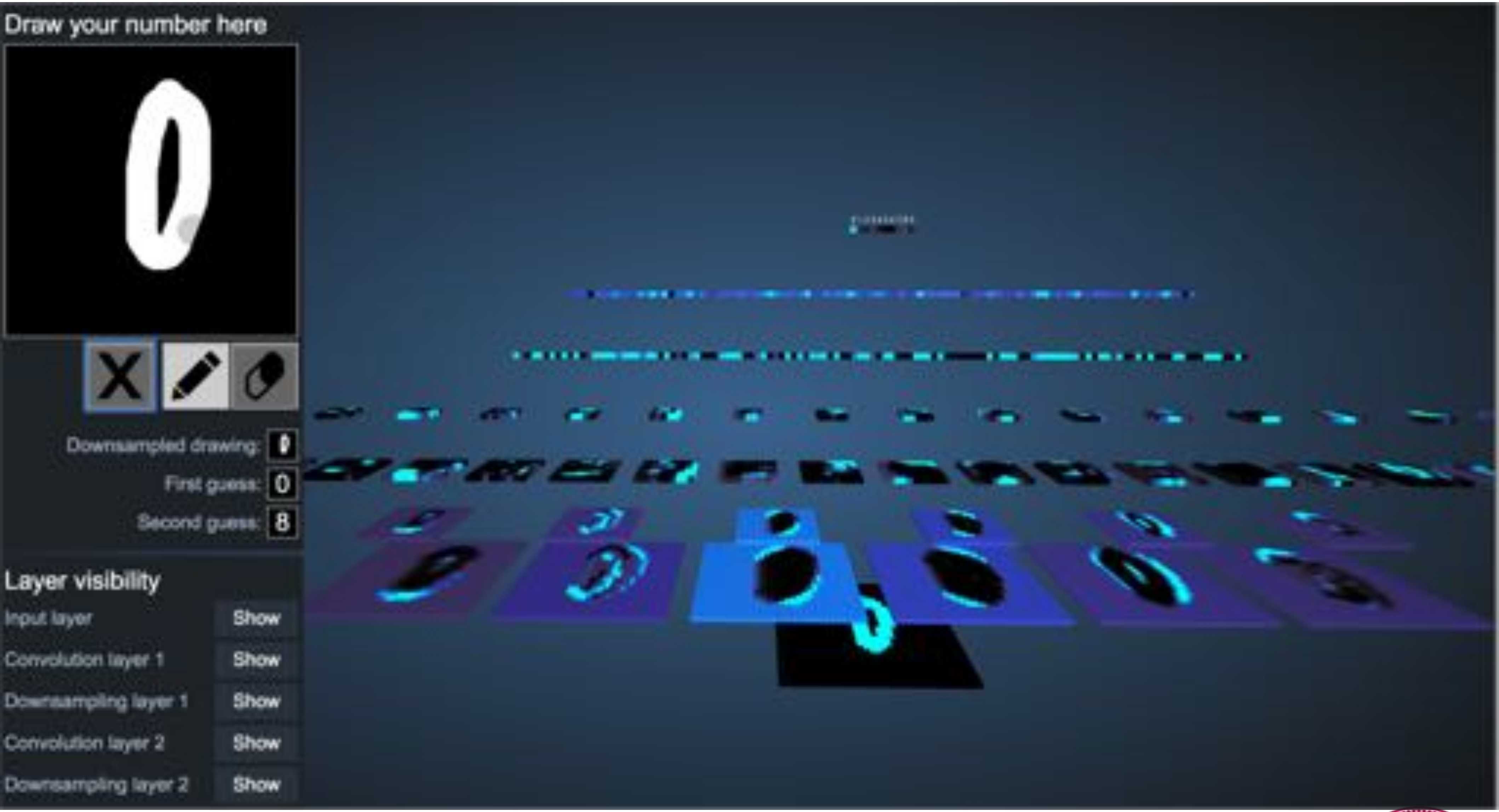


We Can Put Neurons Together



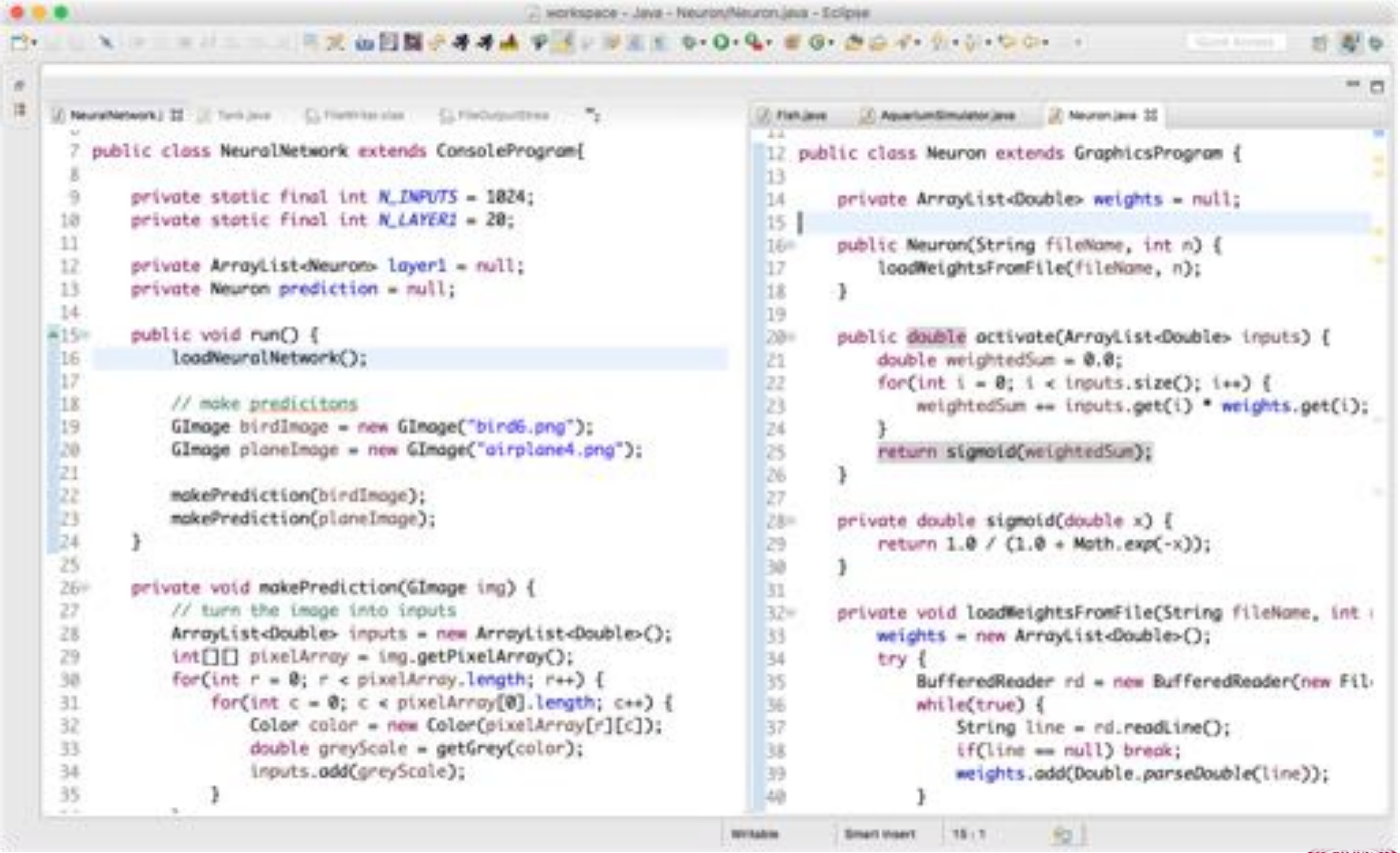
This means it predicts a 0

Demonstration



http://scs.ryerson.ca/~aharley/vis/conv/

What Does This Look Like in Code?



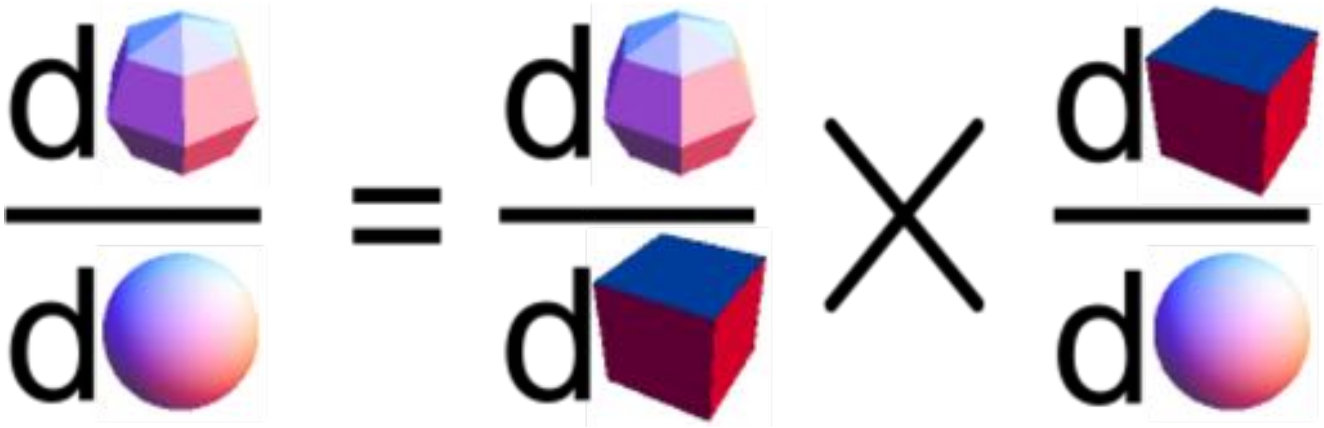
```java
public class NeuralNetwork extends ConsoleProgram{

    private static final int N_INPUTS = 1024;
    private static final int N_LAYER1 = 20;

    private ArrayList<Neuron> layer1 = null;
    private Neuron prediction = null;

    public void run() {
        loadNeuralNetwork();

        // make predicitons
        GImage birdImage = new GImage("bird6.png");
        GImage planeImage = new GImage("airplane4.png");

        makePrediction(birdImage);
        makePrediction(planeImage);
    }

    private void makePrediction(GImage img) {
        // turn the image into inputs
        ArrayList<Double> inputs = new ArrayList<Double>();
        int[][] pixelArray = img.getPixelArray();
        for(int r = 0; r < pixelArray.length; r++) {
            for(int c = 0; c < pixelArray[0].length; c++) {
                Color color = new Color(pixelArray[r][c]);
                double greyScale = getGrey(color);
                inputs.add(greyScale);
            }
```

```java
public class Neuron extends GraphicsProgram {

    private ArrayList<Double> weights = null;

    public Neuron(String fileName, int n) {
        loadWeightsFromFile(fileName, n);
    }

    public double activate(ArrayList<Double> inputs) {
        double weightedSum = 0.0;
        for(int i = 0; i < inputs.size(); i++) {
            weightedSum += inputs.get(i) * weights.get(i);
        }
        return sigmoid(weightedSum);
    }

    private double sigmoid(double x) {
        return 1.0 / (1.0 + Math.exp(-x));
    }

    private void loadWeightsFromFile(String fileName, int 
        weights = new ArrayList<Double>();
        try {
            BufferedReader rd = new BufferedReader(new Fil
            while(true) {
                String line = rd.readLine();
                if(line == null) break;
                weights.add(Double.parseDouble(line));
            }
```

Aside: decomposition

How do we get those weights?

Chain Rule Down the Network



Learning Weights

$$LL(\theta) = y \log \sigma(\theta^T \mathbf{x}) + (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})]$$

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x}]$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

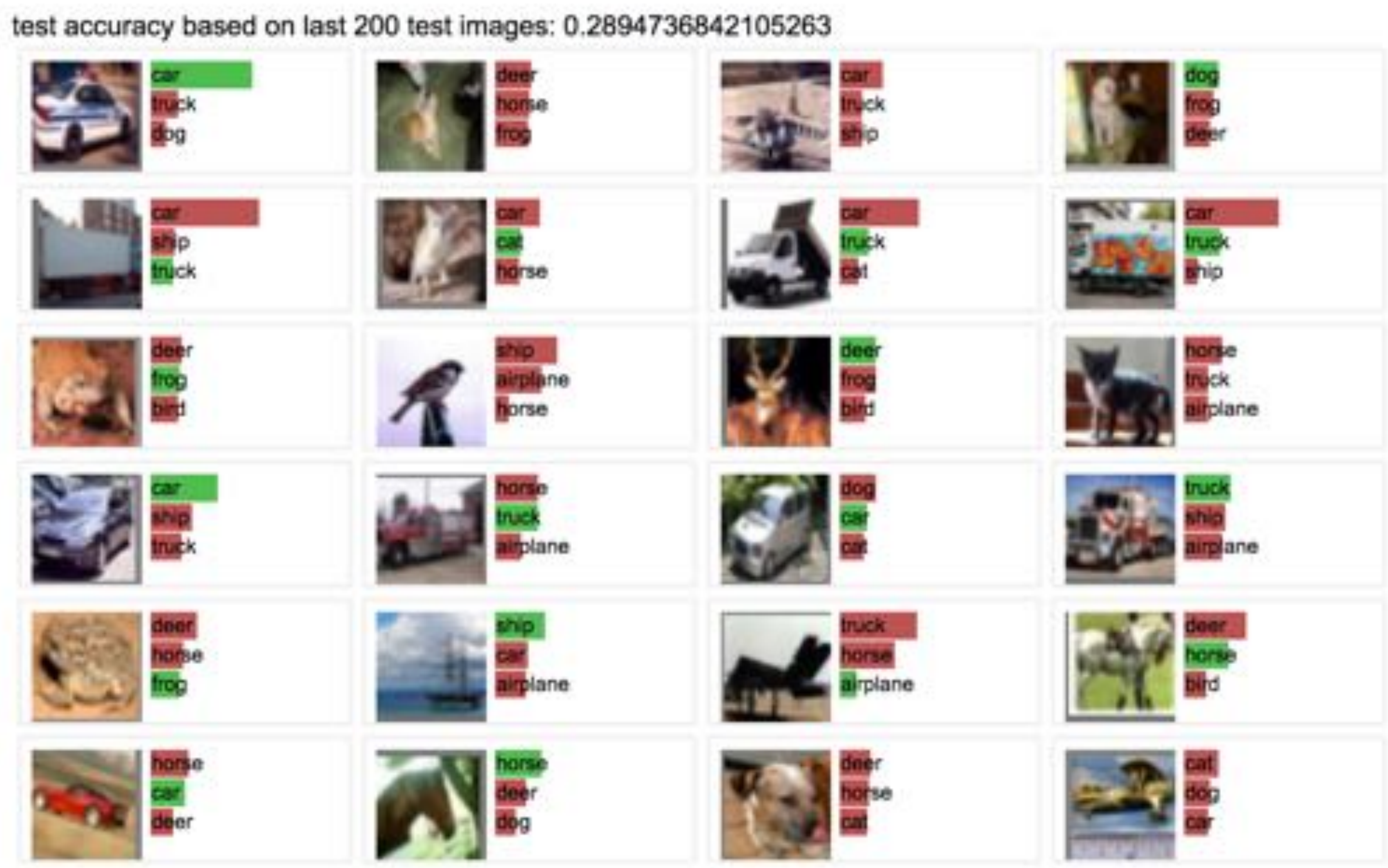
$$= \left[\frac{y - \sigma(\theta^T x)}{\sigma(\theta^T x)[1 - \sigma(\theta^T x)]} \right] \sigma(\theta^T x)[1 - \sigma(\theta^T x)] x_j$$

$$= \left[y - \sigma(\theta^T x) \right] x_j$$

Artificial Neurons: One of the greatest decompositions of our lifetimes

```
model.calculatePartialDerivative(data)
model.update(data)
```

Let’s Train!



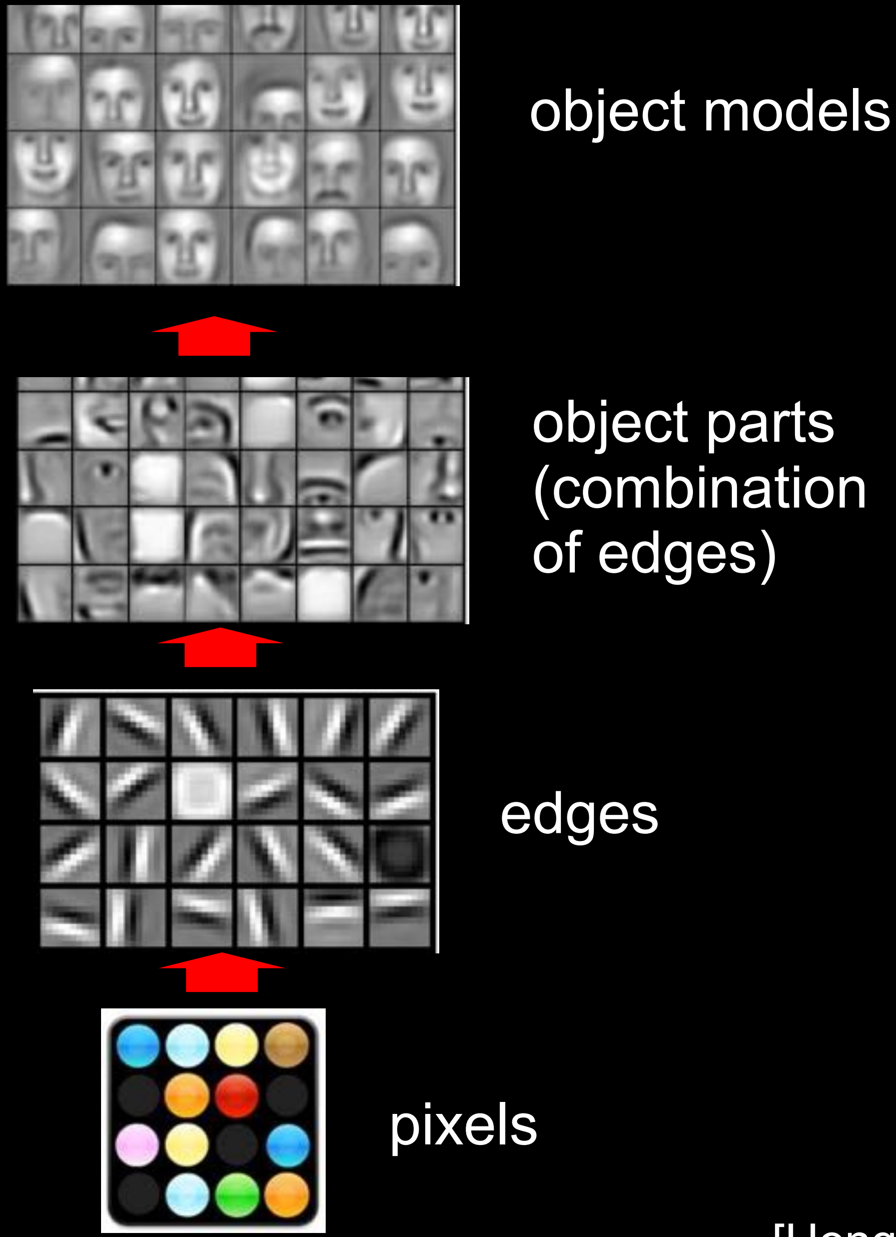
test accuracy based on last 200 test images: 0.2894736842105263

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Like lego pieces

Visualize the Weights

Training set: Aligned images of faces.



object models

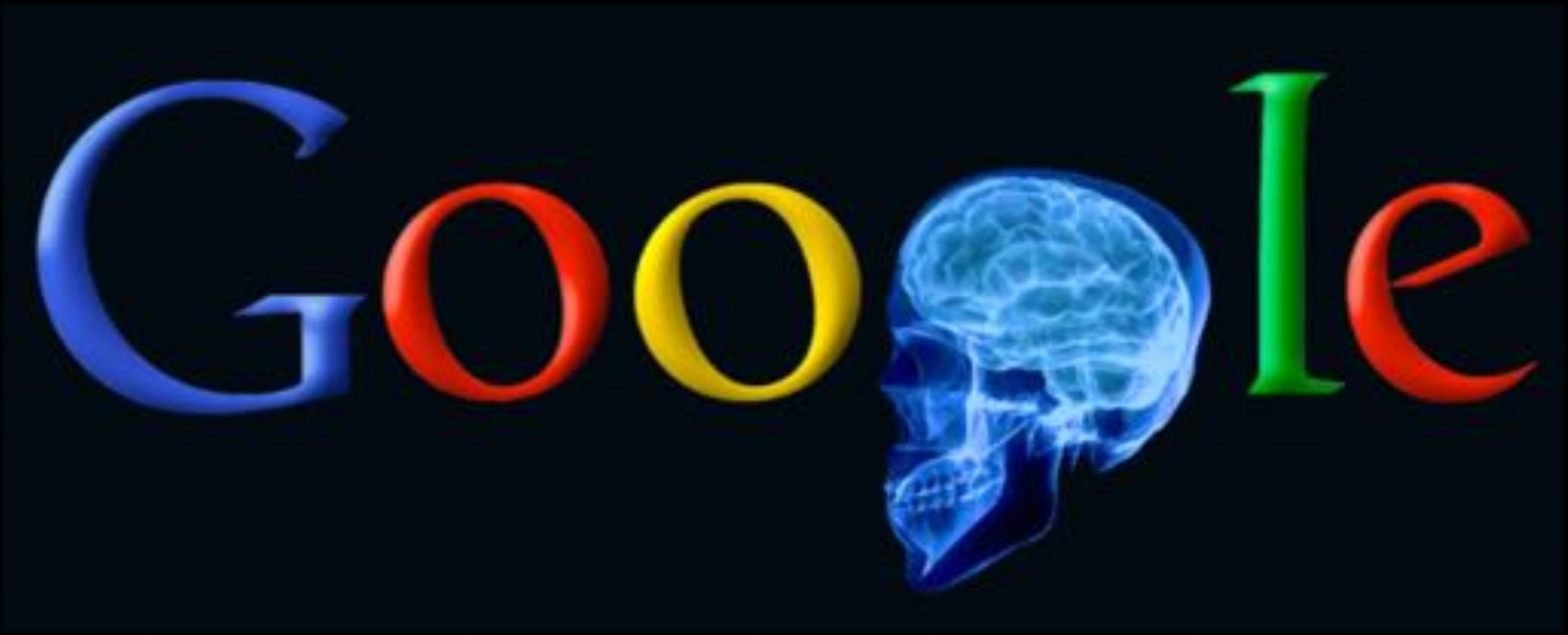
object parts (combination of edges)

edges

pixels

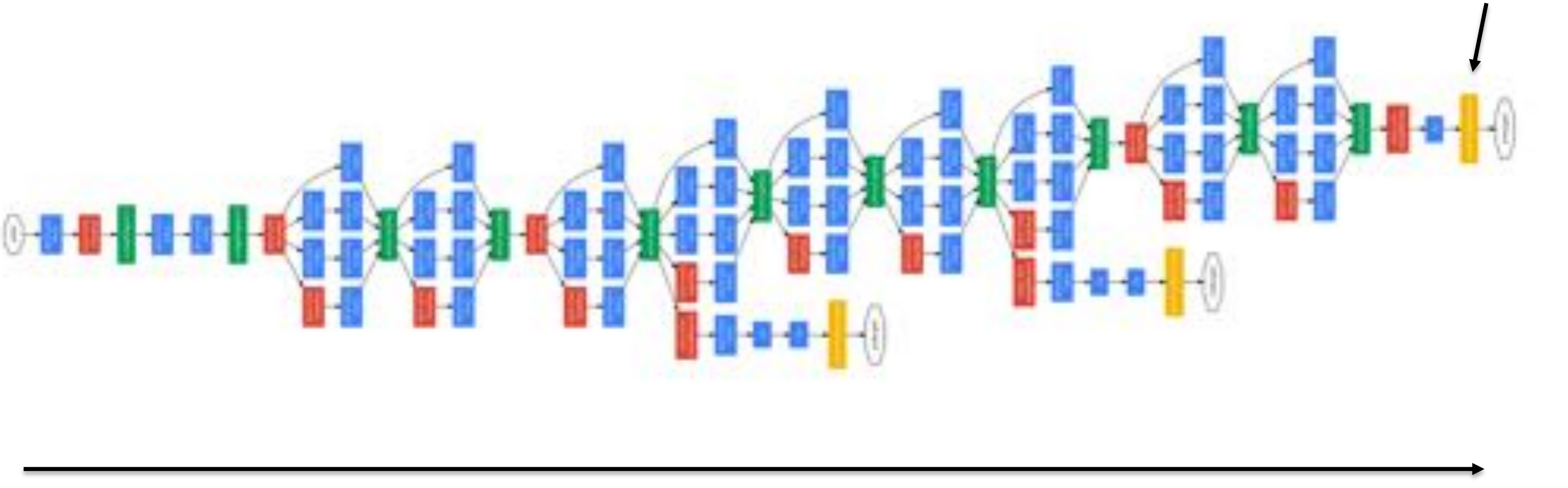
[Honglak Lee]

GoogLeNet Brain



1 Trillion Artificial Neurons

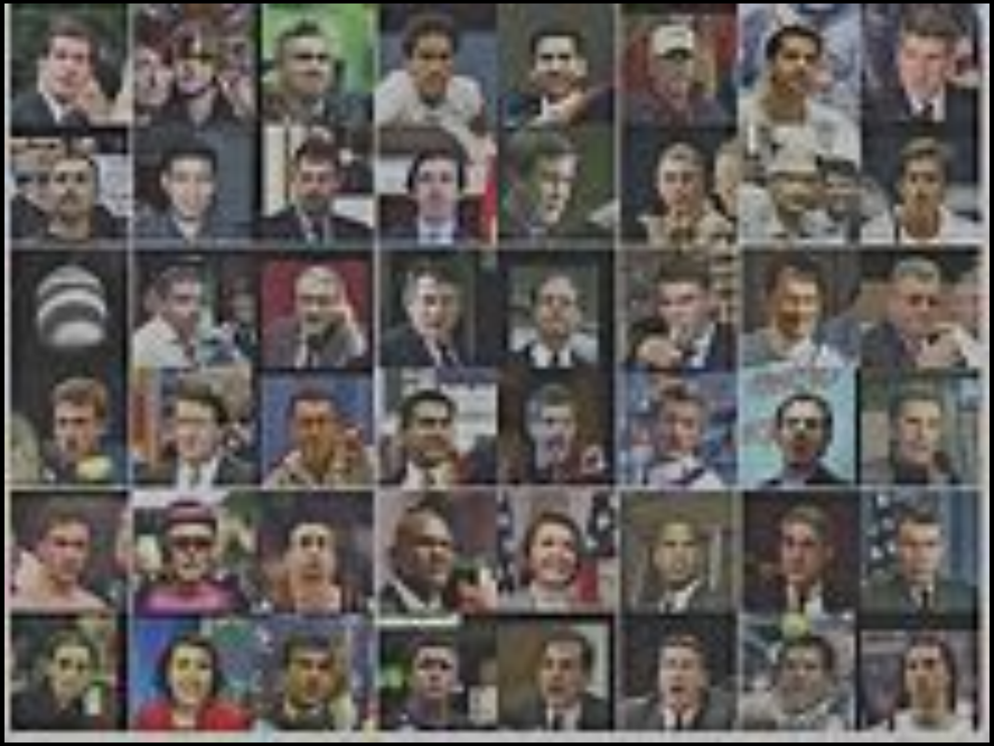
GoogLeNet Brain Graph



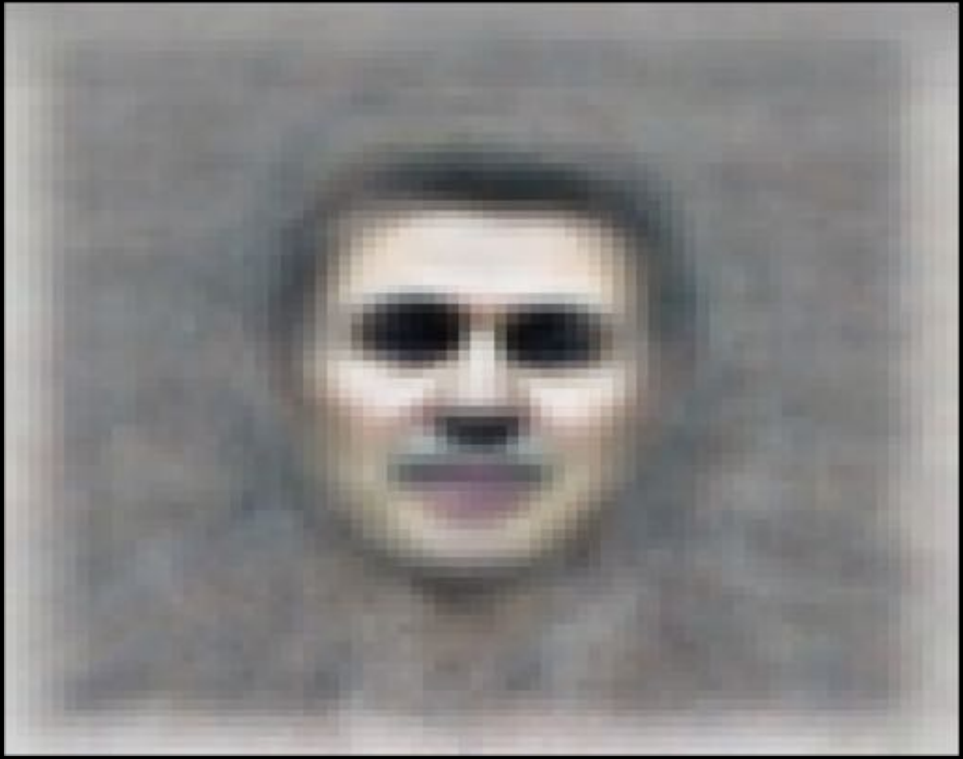
Multiple,
Multi class output

22 layers deep

The Face Neuron



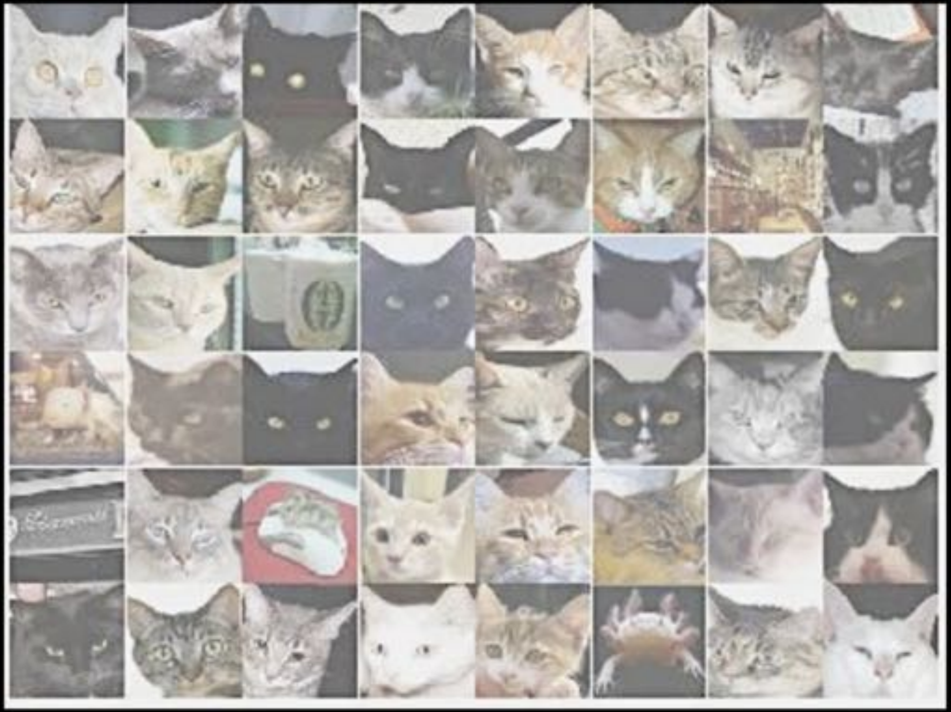
Top stimuli from the test set



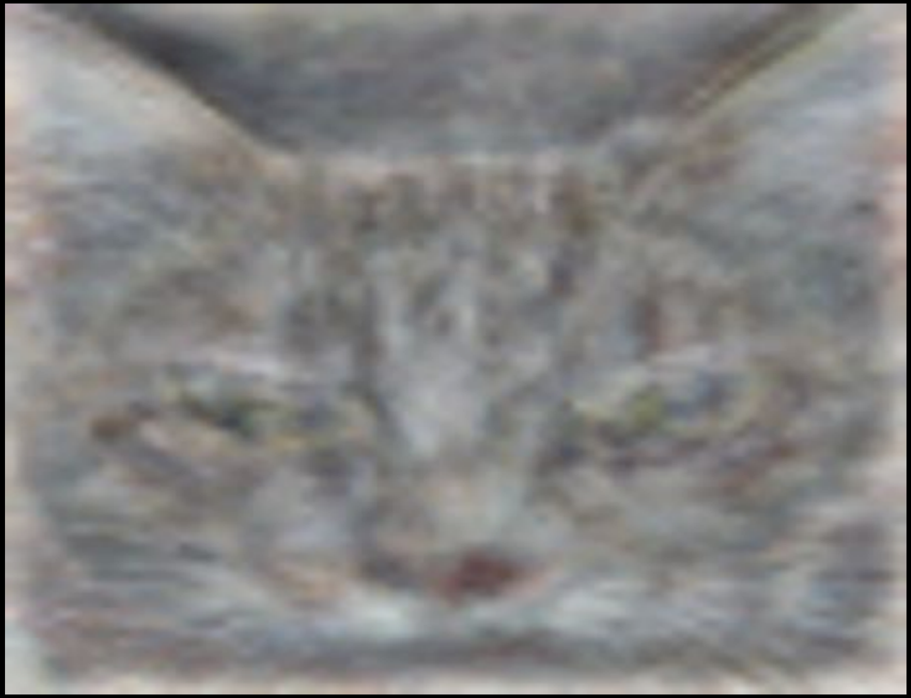
Optimal stimulus
by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

The Cat Neuron



Top stimuli from the test set

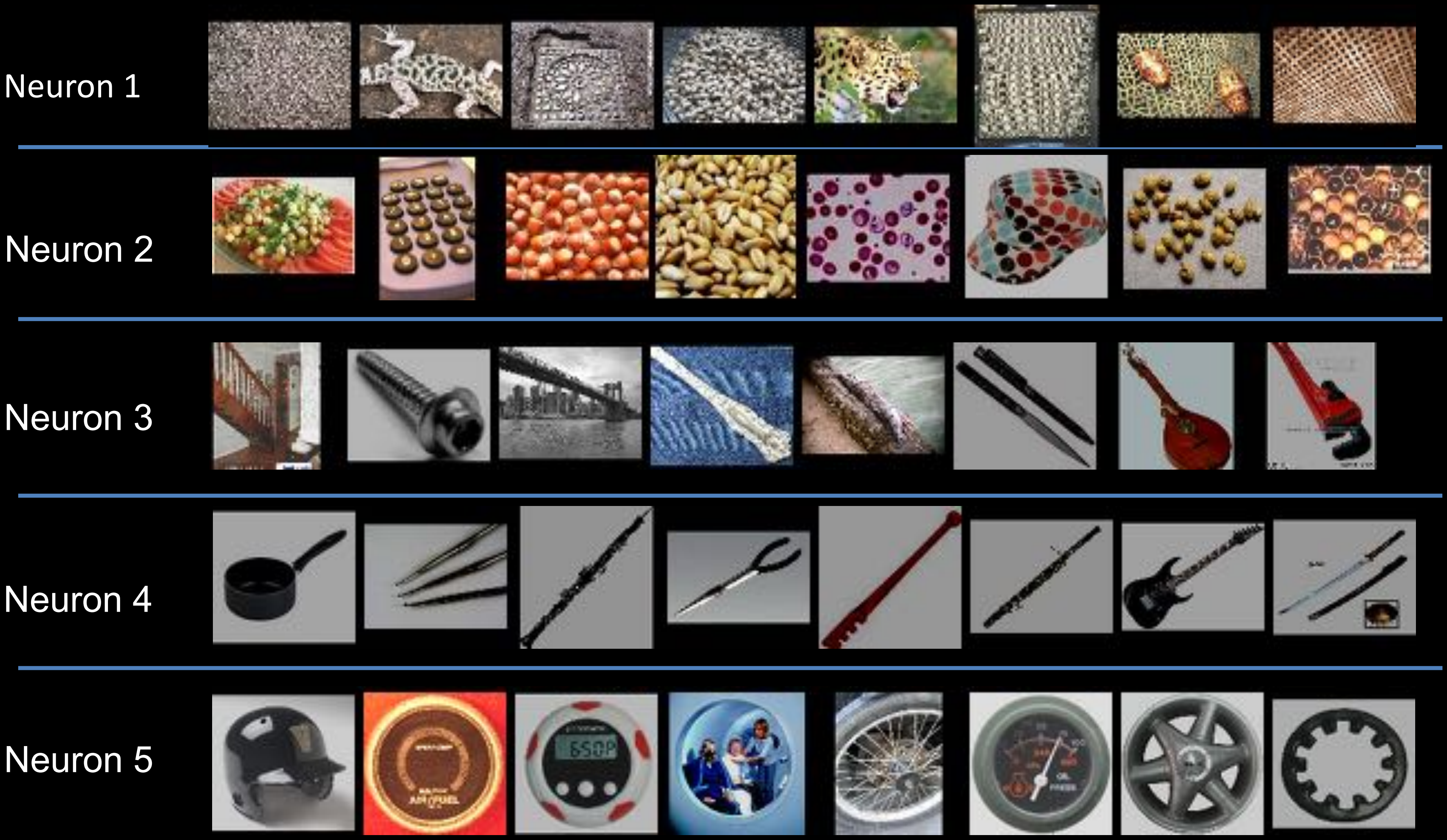


Optimal stimulus
by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

Hire the smartest people in the world
Invent cat detector

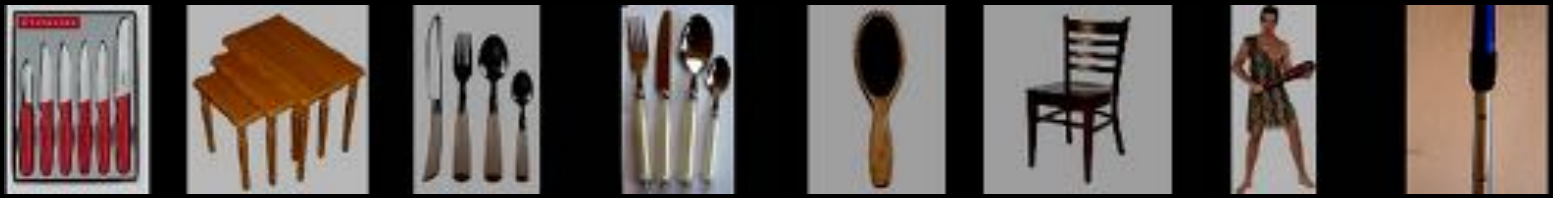
Best Neuron Stimuli



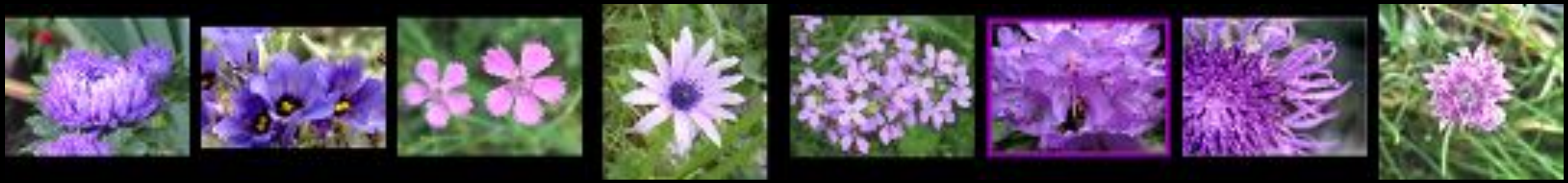
Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

Best Neuron Stimuli

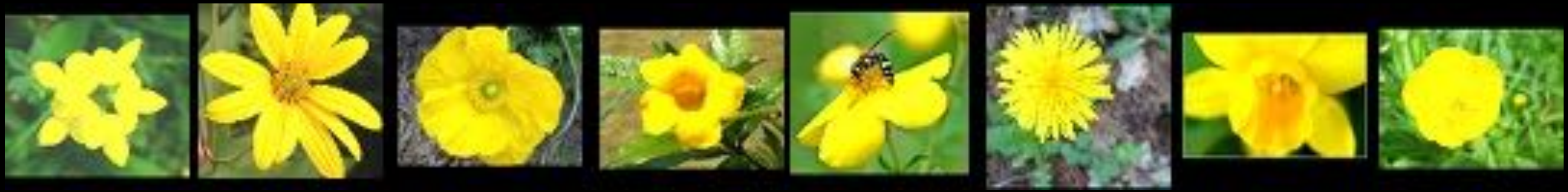
Neuron 6



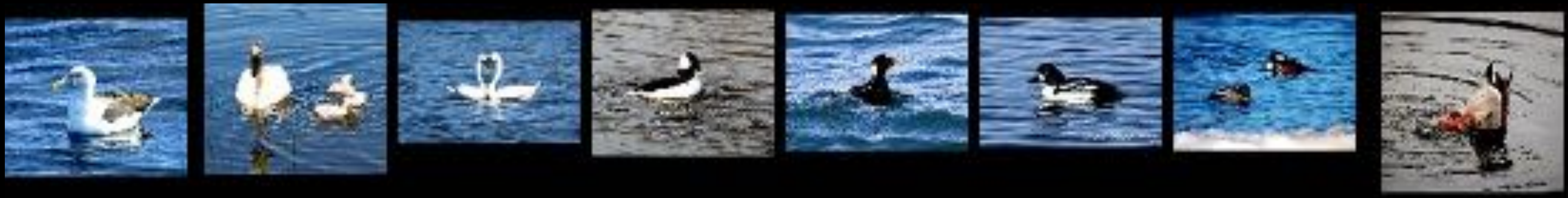
Neuron 7



Neuron 8



Neuron 9



Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

ImageNet Classification

22,000 categories

14,000,000 images

Hand-engineered features (SIFT, HOG, LBP),
Spatial pyramid, SparseCoding/Compression

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

22,000 is a lot!

…
smoothhound, smoothhound shark, Mustelus mustelus
American smooth dogfish, Mustelus canis
Florida smoothhound, Mustelus norrisi
whitetip shark, reef whitetip shark, Triaenodon obseus
Atlantic spiny dogfish, Squalus acanthias
Pacific spiny dogfish, Squalus suckleyi
hammerhead, hammerhead shark
smooth hammerhead, Sphyrna zygaena
smalleye hammerhead, Sphyrna tudes
shovelhead, bonnethead, bonnet shark, Sphyrna tiburo
angel shark, angelfish, Squatina squatina, monkfish
electric ray, crampfish, numbfish, torpedo
smalltooth sawfish, Pristis pectinatus
guitarfish
roughtail stingray, Dasyatis centroura
butterfly ray
eagle ray
spotted eagle ray, spotted ray, Aetobatus narinari
cownose ray, cow-nosed ray, Rhinoptera bonasus
manta, manta ray, devilfish
Atlantic manta, Manta birostris
devil ray, Mobula hypostoma
grey skate, gray skate, Raja batis
little skate, Raja erinacea
…

Stingray

Mantaray

0.005%	1.5%	?
Random guess	Pre Neural Networks	GoogLeNet

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

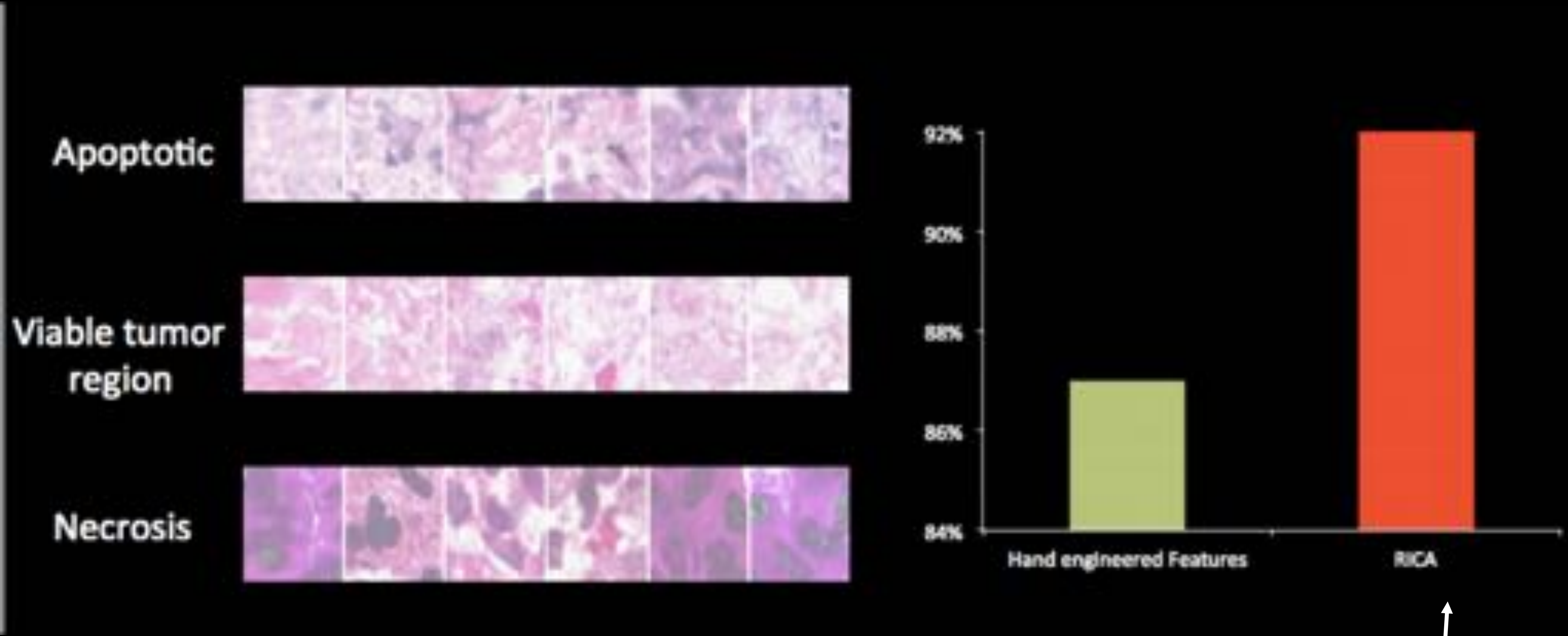
0.005%	1.5%	43.9%
Random guess	Pre Neural Networks	GoogLeNet

Szegedy et al, Going Deeper With Convolutions, CVPR 2015

0.005%	1.5%	82.7%
Random guess	Pre Neural Networks	NASNet

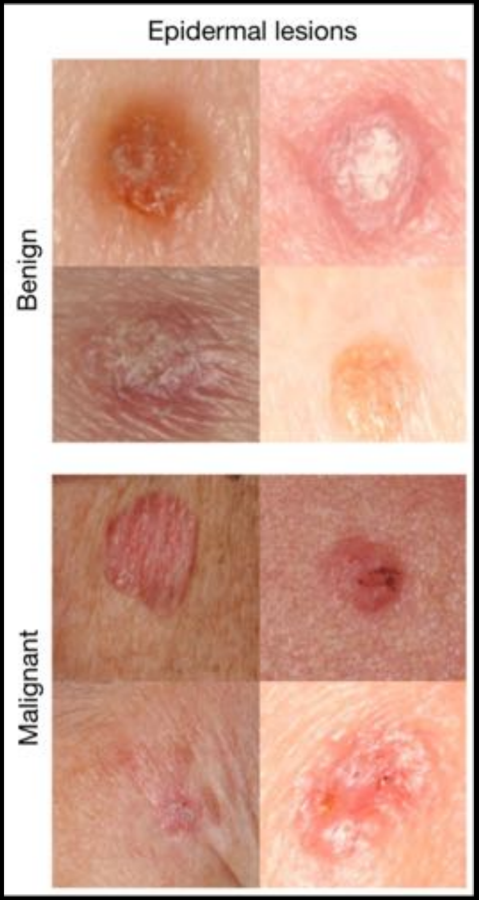
https://arxiv.org/pdf/1707.07012.pdf

Vision has Social Implications



Neural network

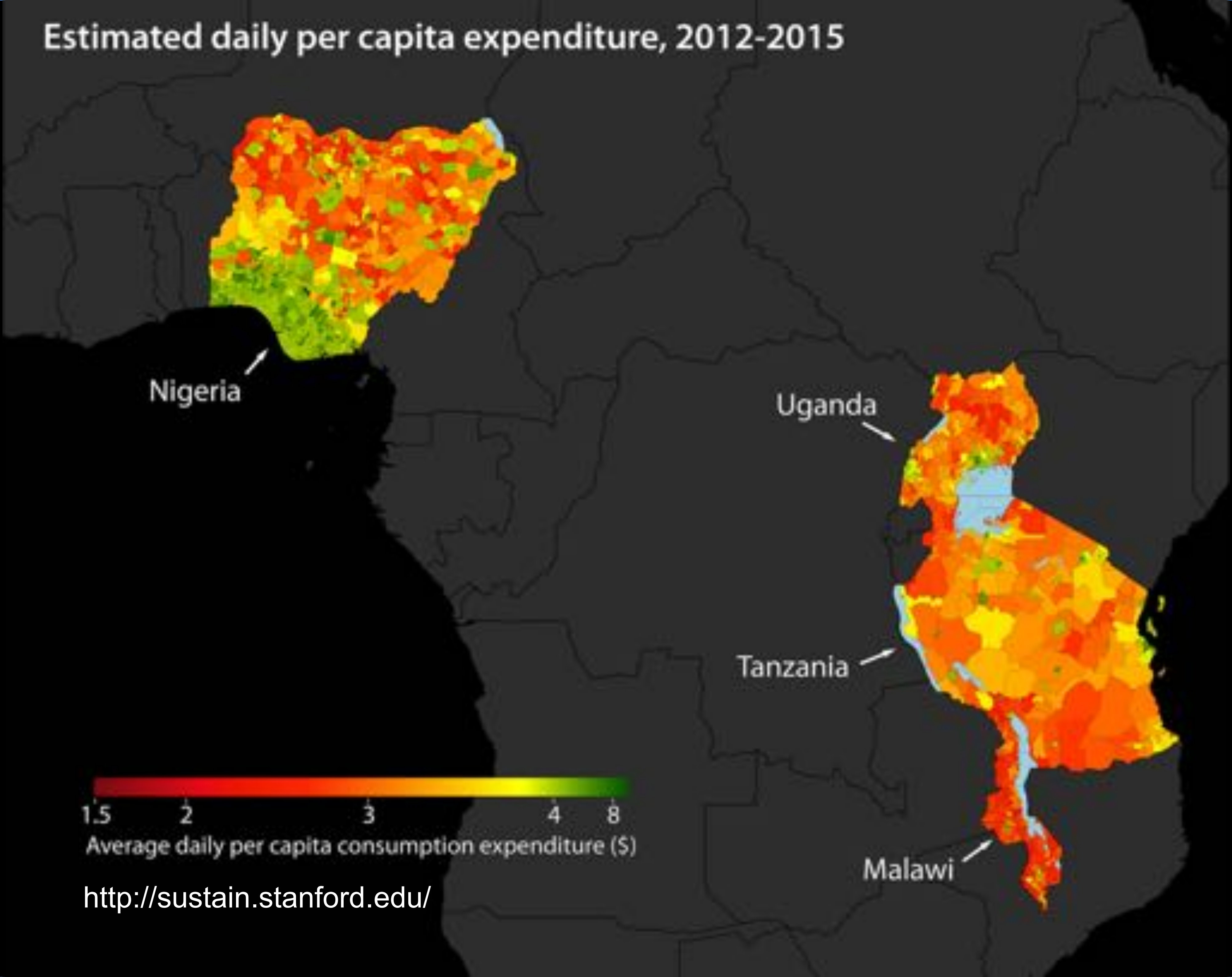
Where is this useful?



A machine learning algorithm performs **better than** the best dermatologists.

Developed this year, at Stanford.

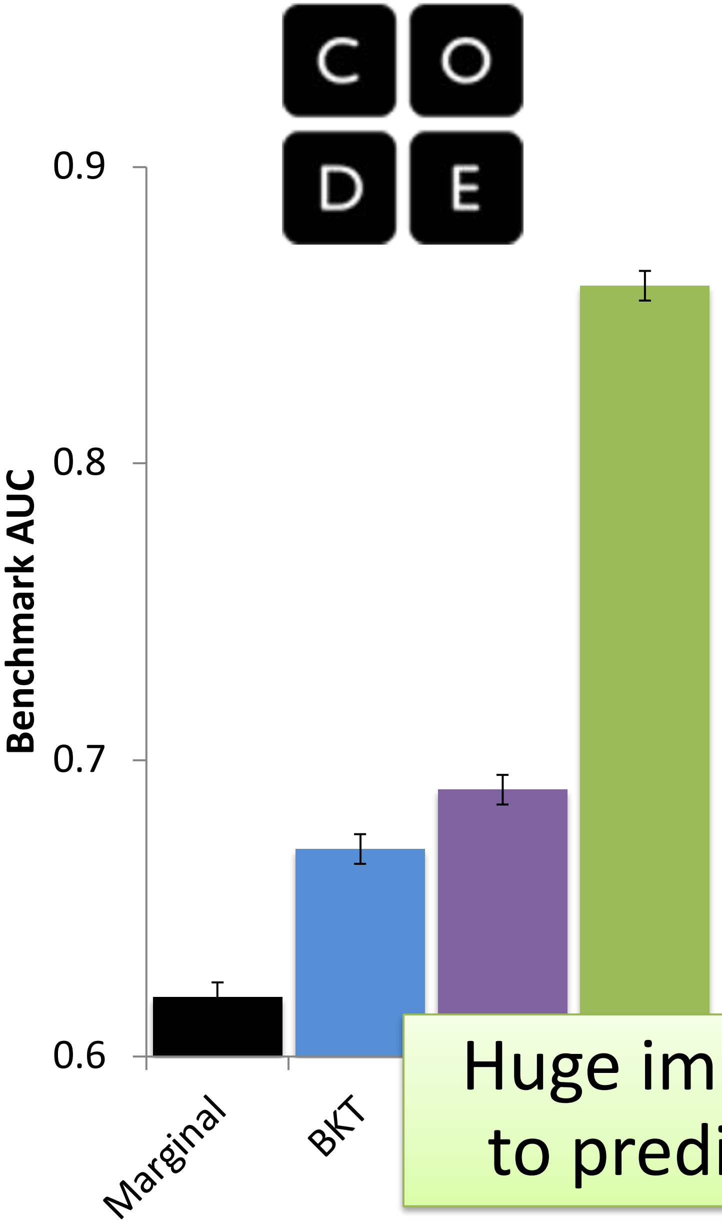
Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

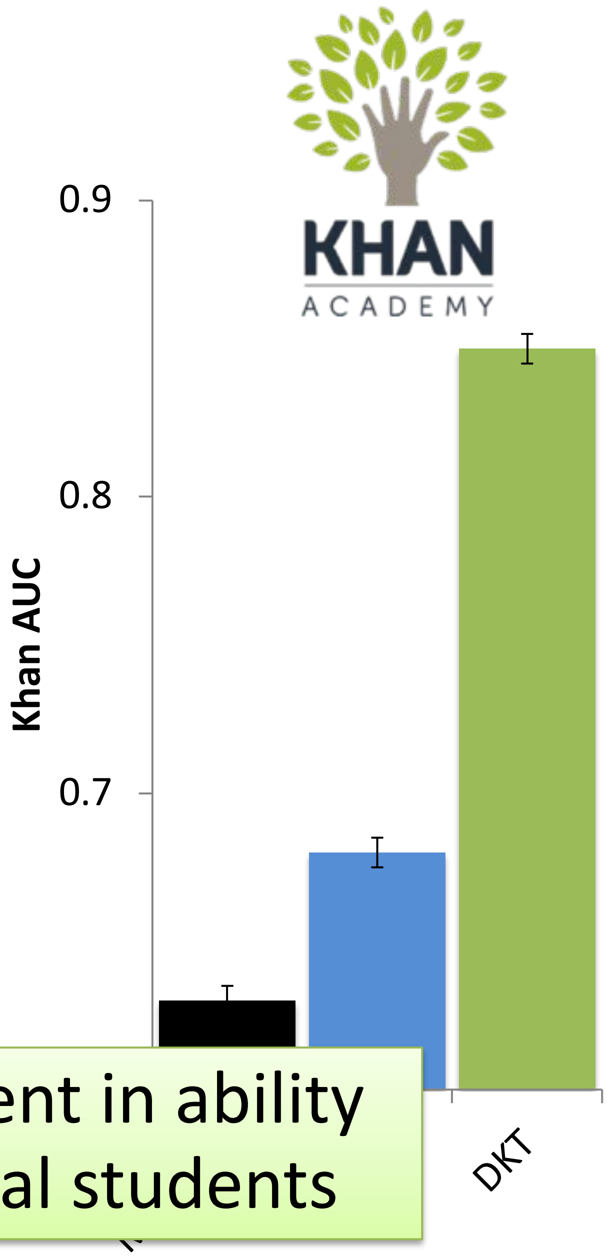


Estimated daily per capita expenditure, 2012-2015

http://sustain.stanford.edu/

Understanding Students



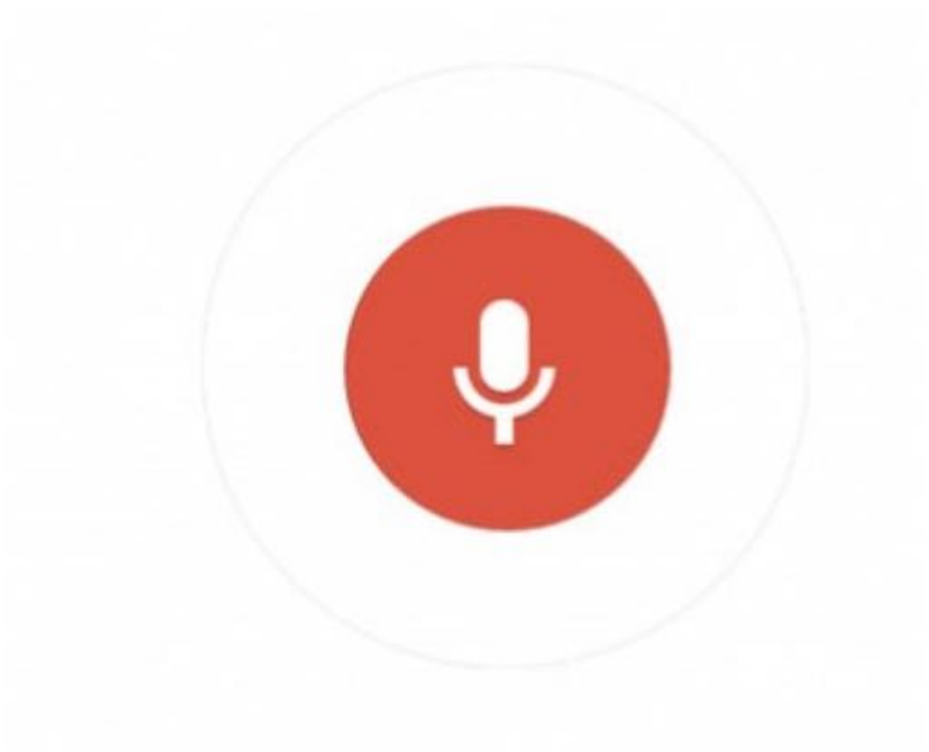


Huge improvement in ability to predict for real students

Tl;dr our brain is constantly decomposing

Told Vision Was 30 Years Out

Told Speech Was 30 Years Out



Almost perfect…

What a time to be alive

Ethics in AI

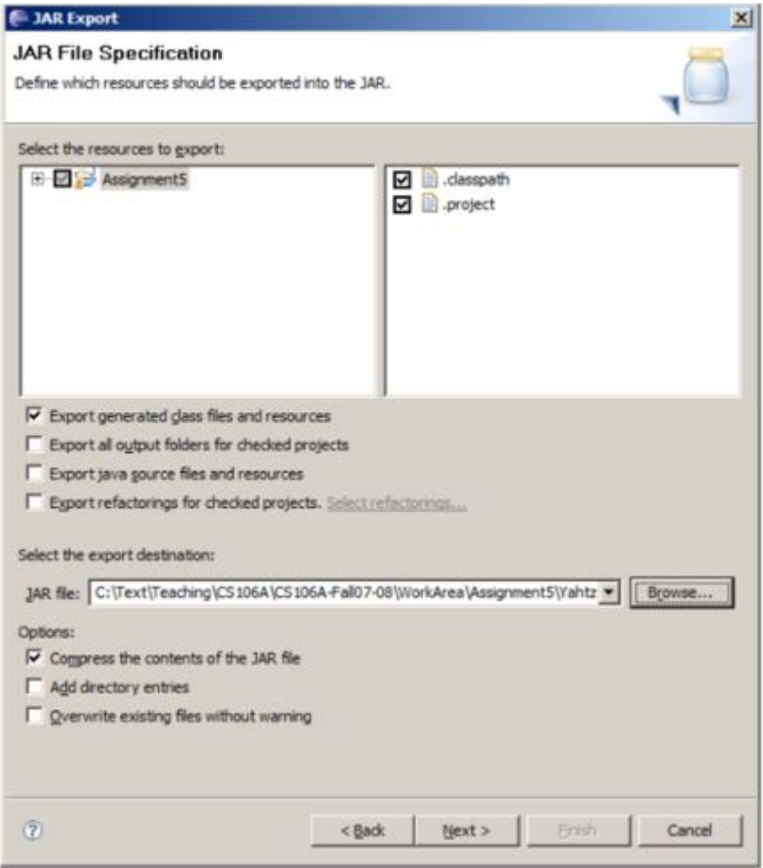
The end

Export to JAR

- **JAR**: Java Archive. A compressed binary of a Java program.
  - The typical way to **distribute a Java app as a single file**.
  - Essentially just a ZIP file with Java

- Making a JAR of your project in E
  - File → Export ... → Java → **Runnable JAR File**

- *see handout on course web site*



Define which resources should be exported into the JAR.

Select the resources to export:

☑ Assignment5

☑ .classpath
☑ .project

☑ Export generated class files and resources
□ Export all output folders for checked projects
□ Export java source files and resources
□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: C:\Text\Teaching\CS106A\CS106A-Fall07-08\WorkArea\Assignment5\Yahtz

Options:

☑ Compress the contents of the JAR file
□ Add directory entries
□ Overwrite existing files without warning